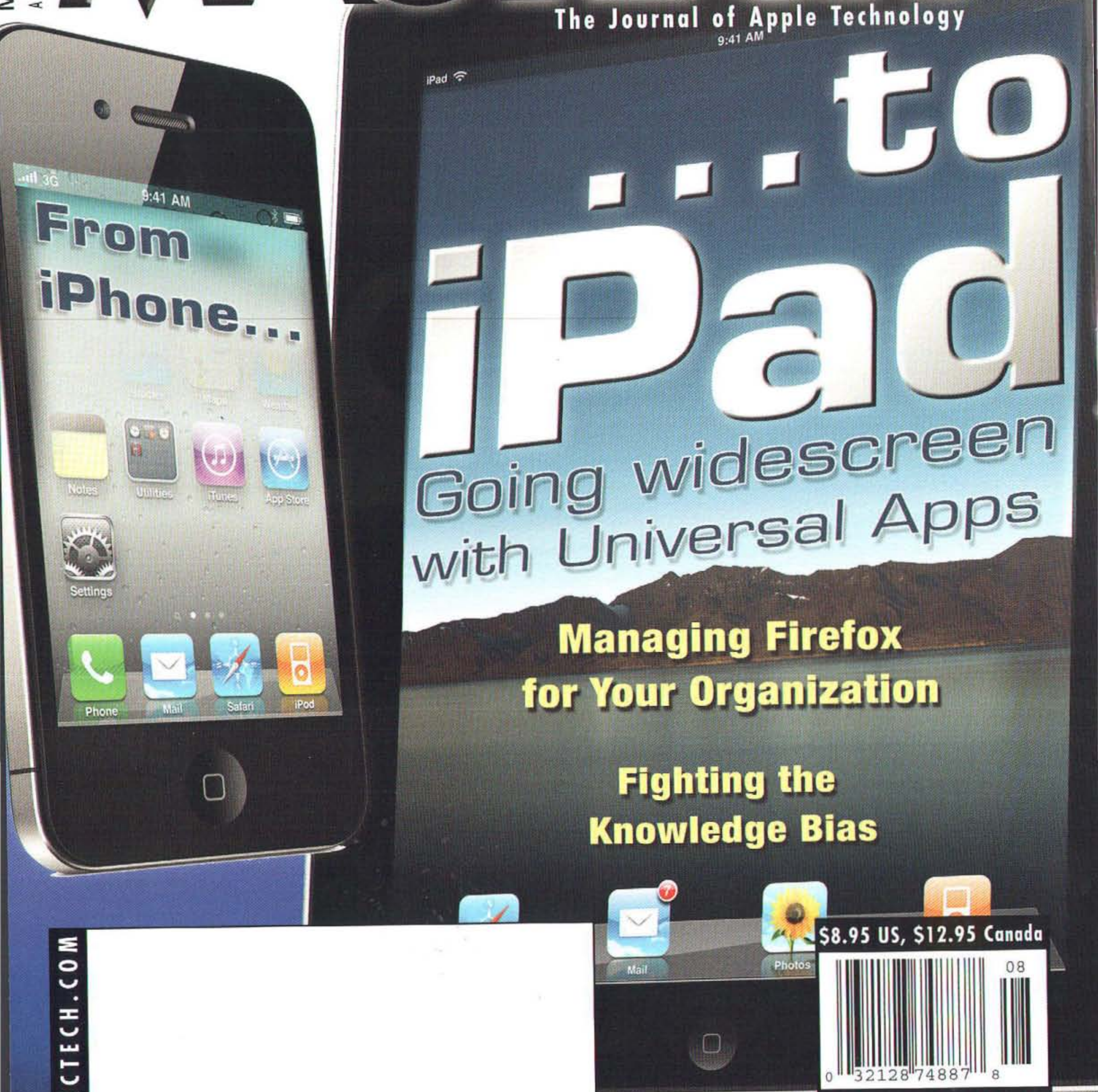
Scripting with SatImage

MacTech Magazine • August • 2010

MACTECH®

The Journal of Apple Technology



Going widescreen with Universal Apps

Managing Firefox for Your Organization

Fighting the Knowledge Bias

MACTECH.COM

$8.95 US, $12.95 Canada

ISSN 1067-8360 Printed in U.S.A.

MACTECH® CONFERENCE 2010

When and Where?

MacTech Conference for IT Pros and Apple developers is November 3-5, 2010, in Los Angeles at the Sheraton Universal in Universal City. The three-day, packed event will have sessions and activities throughout the day and evening giving attendees the opportunity to not only learn from the best, but to also get to know others in the industry.

Two Session Tracks.

The MacTech Conference will have two separate tracks: one focused on IT, and one focused on programming/development. Sessions will focus on both desktop and mobile, with appropriate levels of attention paid to the Mac, iPhone, iPad and iPod.

Packed Schedule. Morning 'til Night.

You won't just be in sessions hearing about great technologies and products. MacTech Conference has a packed evening schedule designed not only to be fun, but also to give you the opportunities to get to know your fellow attendees. This includes an exclusive trip to the world renowned Griffith Observatory, and a party at Jillian's that includes the new MacTech Bowl.

All Meals Included.

This is an immersive conference, and as such, the time you spend with peers you know and new people that you meet is as important as the sessions themselves. We'll be feeding you throughout the event not only to make it all inclusive, but also so that you can maximize your time with other attendees.

Space is Limited.

We have a limited number of conference attendee spots and hotel rooms available. As a conference with hundreds, not thousands, of people, we want you to have time to get to know people. But, that also means that (like other conferences in the Apple market) if you don't act fast, you may miss out.

Subscribers Get Special Pricing.

Everyone could get the early bird pricing in the beginning, but current MacTech subscribers can take advantage of it during an extended early bird window. ***But act fast! Even for subscribers, it ends soon.***

Special registration for current subscribers is:
http://www.mactech.com/conference/subregister

APPRIVER CLIENTS NEVER SEE THIS

Error: Mailbox Full

You have exceeded the storage limit on your mailbox. Delete some important emails or contact your server administrator who is on vacation right now. My neighbor's kid is really good with computers. Maybe he could help you out when he gets home from school. Or call AppRiver to solve this problem now.

Call AppRiver

FREE 30-Day Trial Sign-up NOW

What's New:

Native support for Microsoft® Exchange allows users to integrate their Exchange data into Mac's **Mail, iCal** and **Address Book** applications.

Unlimited Mailbox Storage
No need to manage your mailbox storage quotas.

Spam & Virus Protection
Comprehensive protection against unwanted email.

Entourage & Mac Mail Optimized
Email Performance you would expect on your Mac!

Better Mobility
Synchronize your iPhone with our FREE ActiveSync service, the fastest, most reliable synchronization service available.

appriver® The Messaging Experts™

www.appriver.com/mac

sales@appriver.com

(866) 223-4645

NEW

Switch to Mac has never been easier.

Parallels Desktop® Switch to Mac Edition

Ready. Set. Switch.

- Complete Moving Suite
- Interactive Video Tutorials
- Step-by-Step Easy Migration

Ready. Set. Switch.
Parallels Optimized Computing
Parallels Desktop Switch to Mac Edition
#1 Mac System Utility
Run Windows on Your Mac without Rebooting.
Plus, everything you need to move from Windows to Mac
Stay Current! Includes the latest version of Parallels Desktop for Mac
INCLUDES
✓ Parallels Desktop for Mac
✓ High-Speed USB Transfer Cable
✓ $175 in Bonus Windows Software
"GETTING STARTED ON MAC" VIDEOS
Mac

- Run Windows and Mac Side-by-Side without Rebooting
- Enjoy Your Favorite USB Devices
- Plus $175 Bonus Windows Software

Parallels Desktop Switch to Mac Edition lets you move programs, documents, media and more right from your PC to your new Mac. Then enjoy the best of both worlds and run Windows and Mac OS X side by side.

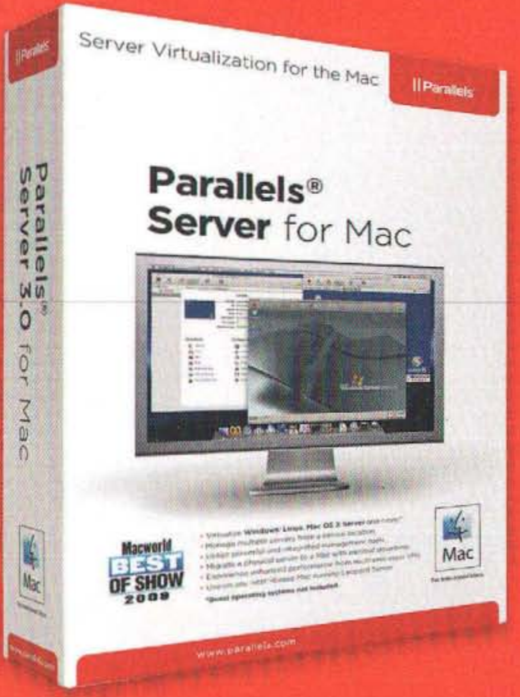
Learn more at
www.parallels.com/products/desktop/stm

Parallels Optimized Computing

Phone +1 (425) 282-6448

|| Parallels™

Parallels® Server for Mac


Server Virtualization for the Mac
|| Parallels
Parallels® Server 3.0 for Mac
Parallels® Server for Mac
Macworld BEST OF SHOW 2009
Mac
www.parallels.com

Make your Mac Server go farther.

Run any OS you choose.

Run any application on the Apple Xserve today.

The world's first server virtualization solution for the Mac platform.

Reduce Server Count by 84%

Save 60% on IT Budget

Oregon City School District

Performance

Full scale hypervisor solution with bare metal architecture.

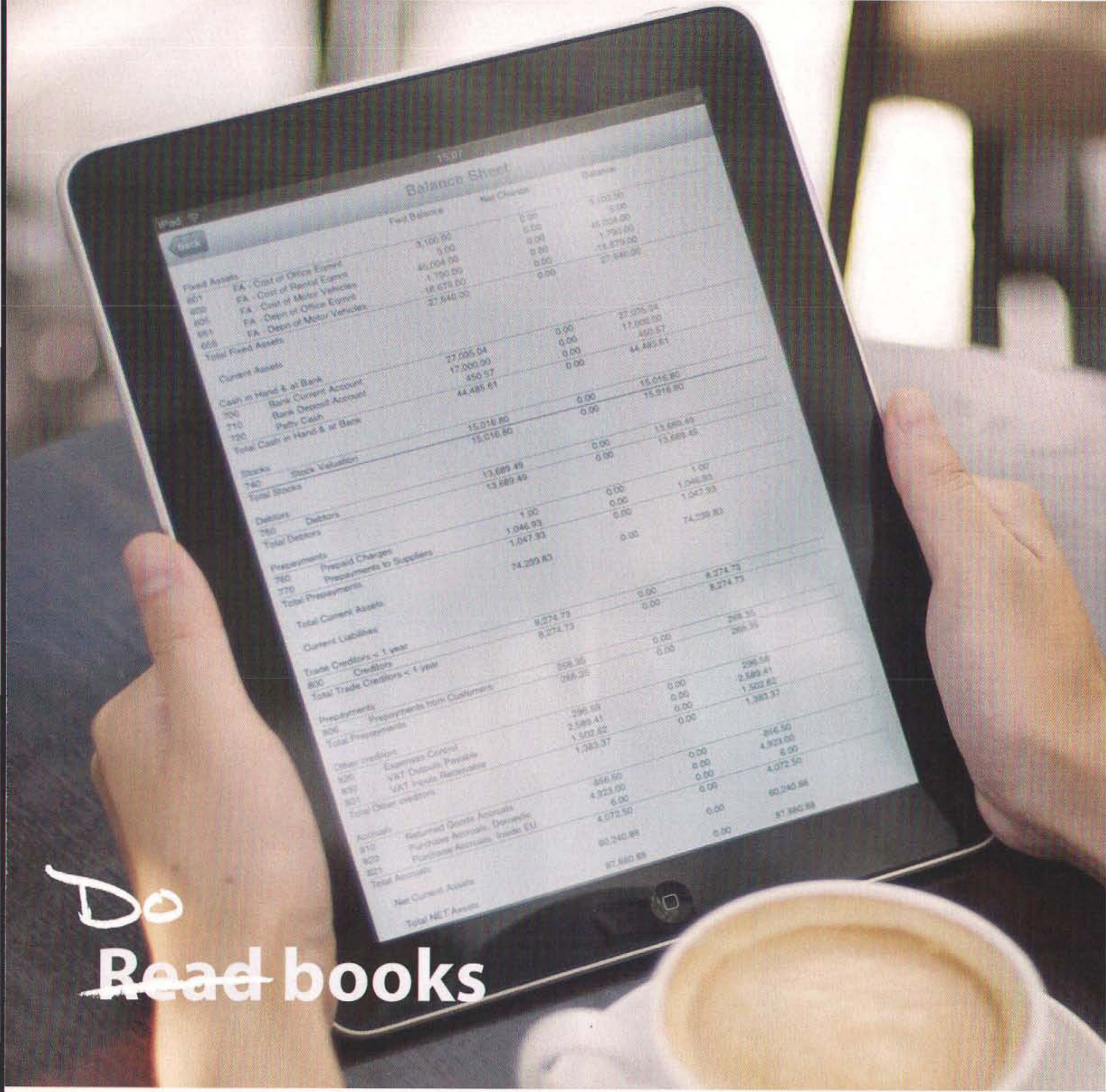
Scalability

Virtualize Mac OS X Server, Windows, Linux and more with support for 32- and 64-bit platform and guest OS support.

Flexibility

Hardware-ready for seamless integration into existing IT infrastructures with built-in VM management and maintenance tools.

Buy now at www.parallels.com or email offer@parallels.com



Accounting | Reporting | Management

Whether you work alone or lead a team of four hundred people,
our solutions will let you bring your business to the Mac.

Books by HansaWorld **Enterprise** by HansaWorld

Integrated CRM | Real-time Mobile | Real-time Consolidation | Integrated Webshop | Built-in Document Management | Powerful Analysis
HansaWorld USA, Inc. 1501 Broadway #12067 & 12068, 12th Floor, New York, NY 10036-5601, USA, Phone: +1 646-459-0603, Email: usa@hansaworld.com

TABLE OF CONTENTS

From the Editor

Coming up in the fall, in November, specifically, is the first ever MacTech Conference. It's a technology conference that is focused to developers and IT practitioners that work mainly on the Mac platform. The reaction has been really great and has been having a bit of a 'halo-effect' on all things MacTech. This month's magazine issue takes advantage of this with authors new and experienced. We balance topics from the 40,000 foot view, right down to the microscopic.

The cover article comes from recurring author (and MacTech Conference speaker) Rich Warren. "From the iPhone to the iPad" covers the issues involved with taking an already-written iPhone app and converting it to the iPad. It sounds easy, but there's really a lot to account for.

New author (and MacTech Conference Speaker) Nathan Toups writes this month about "The Knowledge Bias." As techs, we're sometimes too mired in details and comfortable patterns to really connect with what is going on. Read Nathan's article to find ways to help you combat these tendencies.

Long-time MacTech contributor Mihalis Tsoukalos brings up an important topic: localization. This time, he talks about localizing Dashboard Widgets. You want your software used all over the world, don't you?

José Cruz, another long-time contributor to MacTech writes about Apple-Scripting with SatImage. SatImage is a scripting addition that adds new methods to work with text and functions for math to AppleScript. When you need to push AppleScript a bit, SatImage is an ideal way to wring performance out of AppleScript.

This month's MacEnterprise column by Greg Neagle (also a speaker at this year's MacTech Conference) updates an older column on managing Firefox. This is a particular challenge, as Firefox, in its quest to remain similar in its cross-platform experience, doesn't always do things in a very Mac-like manner. Greg has been through this backwards and forwards at this point, so if you need to manage Firefox in your environment, there's no one better to learn from.

Mike Hjörleifsson brings us all another installment of CoreSec. This month, Mike talks about multi-factor authentication and features an interview with Sven Gossel, CEO of Charismathics, whose company specializes in smart cards and public key infrastructure.

This month's Mac in the Shell column talks about tips and tricks to make working in the bash shell faster and easier. *That* makes you more powerful. These tips are short and manageable—easy to add into your repertoire one at a time.

In the MacTech Spotlight, we feature Jayson Adams from Circus Ponies Software. You're using Circus Ponies award-winning organization tool Notebook, right? Jayson's history goes beyond traditional Mac development. Check out what brought him to where he is now in this month's MacTech Spotlight.

Thanks to all of the loyal readers of MacTech for keeping MacTech *the* journal for Macintosh and Apple technology. We hope to see you at MacTech Conference in November (http://www.mactech.com/conference). Until then, see you next month right here in print.

Ed Marczak,
Executive Editor

Communicate With Us

Department E-Mails

Orders, Circulation, & Customer Service
cust_service@mactech.com

Press Releases
press_releases@mactech.com

Ad Sales
adsales@mactech.com

Editorial
editorial@mactech.com
(Authors only, no pr)

Accounting
accounting@mactech.com

Marketing
marketing@mactech.com

General
info@mactech.com

Web Site
http://www.mactech.com

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?**

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact MacTech Magazine Customer Service at 877-MACTECH

We love to hear from you! Please feel free to contact us with any suggestions or questions at any time.

Write to letters@mactech.com or editorial@mactech.com as appropriate.

MACTECH®

The Journal of Macintosh Technology

A publication of **XPLAIN** CORPORATION

The Magazine Staff

Publisher & Editor-in-Chief: Neil Ticktin
Executive Editor: Edward R. Marczak
Business Editor: Andrea Sniderman
Ad Director: Bart Allan
Production: David Allen
News: Dennis Sellers
Podcast Producer: Josh Long
Staff Writer: Chris Tangora
Staff Writer: Frank Petrie
drupalmaster: Erik Peterson

Xplain Corporation Senior Staff

Chief Executive Officer: Neil Ticktin
President: Andrea J. Sniderman
Accounting: Marcie Moriarty
Customer Relations: Susan Pomrantz

Columnists

Mac In The Shell: by Ed Marczak
The Road to Code: by Dave Dribin
Swaine Manor: by Michael Swaine
KoolTools/Geek Guides: by Dennis Sellers
MacEnterprise: by Philip Rinehart and Greg Neagle

Regular Contributors

José R.C. Cruz, Michael Göbel, Michele Hjörleifsson, Mihalis Tsoukalos
Oliver Pospisil, Rich Morin, William Smith

Canada Post: Publications Mail Agreement #41513541
Canada Returns to be sent to: Bleuchip International, P.O. Box 25542, London, ON N6C 6B2

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 705 Lakefield Road, Suite I, Westlake Village, CA 91361. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

by Edward Marczak

Five Favorite Shell Tricks

My favorite ways to make bash easier and more powerful

Welcome

Once you start using the bash shell, you tend to stick with it as you realize its power and flexibility. However, I often see people that have been using bash for a short period of time—they're not exactly *new* to it, but neither do they have man-months of real usage—that could be using the shell a little more efficiently. Some of this is simple personal preference. However, some of these tips, once first seen quickly turn into can't-live-withouts. Presented without delay, here are my 5 favorite customizations for bash that make one's work easier.

First: History Modifications

In short: Work in a shell involves typing. A lot of it. All of it is recorded in the shell's history file. It turns out that a lot of what you type is somewhat repetitive. A few small tricks can see you building on the lines of built-up history and typing a lot less. bash retains a history of each command you type. Learning to work with—and exploit—this history is crucial to creating a more streamlined experience.

Adding Date and Time

Why: To know *when* you ran that command.

How: Change the HISTTIMEFORMAT shell variable. Add it to your ~/.bash_profile to make it take effect at each invocation of bash. Add this line into your ~/.bash_profile startup:

```
export HISTTIMEFORMAT='%b %d %H:%M:%S: '
```

Yields:

```
5923  Jul 28 13:25:13: sudo make install
5924  Jul 28 13:25:53: type node
5925  Jul 28 13:26:10: vi hello.js
5926  Jul 28 13:29:30: node hello.js
```

Explanation: By default, bash just records the commands typed at the command line in order, but without a timestamp. Often, you can remember that you'd like to repeat something you did (or need to recall how you did) in bash "last Monday evening." In contrast, a typical history listing looks like this:

```
$ history
    1  ps ax
    2  cd /var/log
    3  ls -laort
    4  tail -f system.log
```

Having a timestamp associated with each command makes it much easier to trace your work. You may even consider putting this in the system-wide /etc/profile to affect all bash users.

Modifying Up Arrow Behavior

Why: Modify the up and down arrow key binding to search history to more quickly zero-in on a specific, recently used command.

How: Perform the following key bindings, and add them to your ~/.bash_profile to run the commands at each invocation of the bash shell (to make them 'stick'):

```
bind '"\e[A"':history-search-backward
bind '"\e[B"':history-search-forward
```

Explanation: The default key binding provided in a stock OS X system maps the up and down arrow to step through the history list in order, regardless of what is on the command-line. With this new binding in place, you can type the first few letters of a command, press the up-arrow key and find the most recent match. Press it again to find the next most recent match. Try it, you'll like it.

Using Ctrl-R to Search

Why: to locate a specific command in history.

How: Press Ctrl-R and start typing.

Explanation: Editing in bash defaults to emacs mode, and ctrl-r is a 'reverse search' in emacs. This will not work if you, or someone else, runs bash in vi-mode. If you want to find the last time you ran a command that contained a particular word—a hostname, perhaps—performing a ctrl-r reverse search through history makes your life much easier. You can even press ctrl-r *again* after finding a match to cycle to the previous match, and so on. This is different from the previous tip, "Modifying Up Arrow Behavior," in that you can type and match any part of the command, not just the beginning.

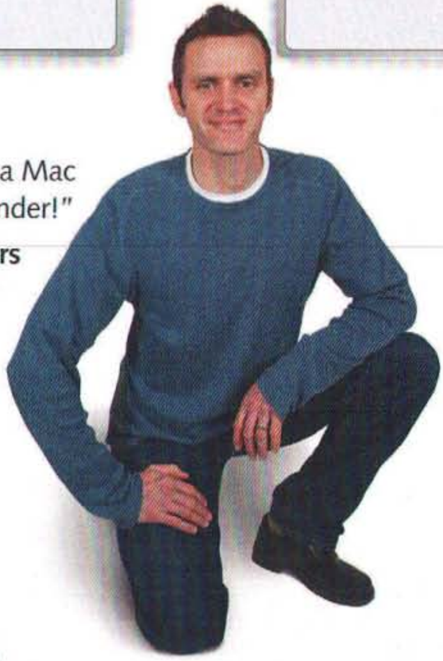
Second: bash Completion

In short: similar to the first tip ("History Modifications"), bash completion will help you type less by inferring what you mean. Well, OK: it's not magic; it can't read your mind. However, bash can pass your current command off to completion functions that can try to complete your typing for

Type Less. Create More.

Effortlessly insert frequently-used words, phrases and graphics, by typing short abbreviations. Great for email, blogging and more!

`thx`	Thanks for your interest in our creative services.
`chk`	Check out our portfolio at: **www.lapis-design.com**
`mysig`	Michael Fetters *Lapis Design*



"I can't work on a Mac without TextExpander!"

Michael Fetters
Mac user

textexpander

Now Available!
TextExpander touch™ for iPad, iPhone & iPod touch

Integrate TextExpander touch in your apps with our free SDK:
www.smileonmymac.com/sdk

Smile on my mac

disclabel PDFpen pagesender textexpander you. It's simple: just press the tab key to invoke this behavior.

Why: To let bash type for you and speed up your interaction with the shell.

How: Tab-completion is built into bash, however, it's a bit limited and a bit, shall we say, stupid. You'll need to install sensible bash-completions. Fortunately, there's a great base to start from: Ian McDonald's bash_completion. There are two easy options to get the required file. Either:

A) Download http://www.caliban.org/files/bash/bash-completion-20060301.tar.gz

or

B) Use MacPorts to install "bash-completion"

In either case, it's highly recommended to add completion to the system-wide profile. If you grabbed the tarball directly (option "a"), first, untar it:

```
tar xzvf bash-completion-20060301.tar.gz
```

Second, copy the contents of the bash_completion.sh file onto the pasteboard:

```
cat bash_completion/bash_completion.sh | pbcopy
```

...and paste that into /etc/profile (using your favorite text editor, of course). Finally, copy bash_completion and bash_completion.d into /etc:

```
sudo cp -pv bash_completion/bash_completion /etc/
sudo cp -Rpv bash_completion/bash_completion.d /etc/
```

...and you're done. The next shell you open will have the advanced completion.

If you installed bash_completion via MacPorts, you really only have one step: source bash_completion in /etc/profile. Add the following line to /etc/profile (or your personal ~/.bash_profile) using your favorite text editor:

```
. /opt/local/etc/bash_completion
```

(note the leading dot character in that line). You're done. The next shell you open will have the advanced completion

Yields: Intelligent tab completion. Open Terminal.app and your present directory will be home. Type "cd Si" (without quotes) and press tab: the command will automatically be completed to "cd Sites/". Now, type "touch Site" and repeat the tab completion. It still works as expected, right? You get "cd Sites/". Prior to our more intelligent tab completion, though, the completion for "cd" wasn't smart enough to know the difference between something it can use—the directory "Sites"—and something it can't use—the file "Site".

Also notice that you can now auto-complete hosts for ssh. The list is derived from your known_hosts file, so, you'll

Make Instant Connections

Sonnet ExpressCard/34 Adapters

Take advantage of your notebook's ExpressCard® slot and expand beyond the standard ports! Get connected fast with Sonnet—from pro dual-port eSATA controller, dual-slot CF card reader, and FireWire® 800 adapter, to GigE adapter, multi-format memory card readers, USB adapters and SATA controllers, our cards offer best-in-class performance. Connect your Mac® or Windows® notebook to a world of possibilities.

sonnettech.com/computercards

SONNET™
SIMPLY FAST™

Sonnet ExpressCard/34 adapters from left to right: Pro Dual CompactFlash, 4-Port USB 2.0, Tempo SATA edge, FireWire 800 Pro, FireWire/USB, Presto Gigabit Ethernet, 21-in-1 Multimedia Memory Card Reader & Writer, Tempo SATA, Tempo SATA Pro.
©2010 Sonnet Technologies, Inc. Sonnet, the Sonnet logotype, and Simply Fast are registered trademarks of Sonnet Technologies, Inc. FireWire and MacBook are trademarks of Apple, Inc., registered in the U.S. and other countries.

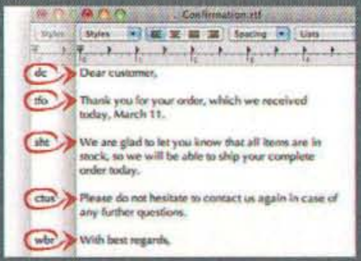
Work less. And finish faster.*

Let us do the tedious work for you.

Typing the same text over and over?

Typinator

types frequently used text for you and auto-corrects your typos.



dc → Dear customer,

tfo → Thank you for your order, which we received today, March 11.

sht → We are glad to let you know that all items are in stock, so we will be able to ship your complete order today.

ctus → Please do not hesitate to contact us again in case of any further questions.

wbr → With best regards,

Want to drive your Mac like a Pro?

KeyCue

helps you speed up your daily tasks with keyboard shortcuts.

Need special characters in your document?

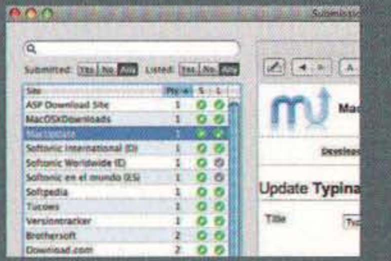
PopChar

inserts any special character into your document with just two clicks.

Have great software to sell?

Shareware Publisher

submits your software products to leading download sites.



Site	Pty	S	L
ASP Download Site	1	●	●
MacOSXDownloads	1	●	●
MacUpdate	1	●	●
Softonic International (D)	1	●	●
Softonic Worldwide (E)	1	●	●
Softonic en el mundo (ES)	1	●	●
Softpedia	1	●	●
Tucows	1	●	●
Versiontracker	1	●	●
Brothersoft	2	●	●
Download.com	2	●	●

www.ergonis.com/mactech

* **Caution:** Usage of Ergonis productivity boosters will revolutionize the way you use your Mac and make you more productive. You will have to look for another hobby!

have had to have accessed them at least once before (or manually added the host to your known_hosts file).

In the completions file you just installed, you've enabled completions for git, tar, tcpdump and more. When in doubt, press tab—you may be pleasantly surprised.

Readline and Editline

Recently, I was schooled in the readline library on Mac OS X. Since I've used readline functions under OS X, I asserted that readline was a part of OS X. Well, pedantically speaking, readline *isn't* a part of OS X.

The readline library is licensed GPL, which is a problematic license for many, including Apple. There is a substitute library named editline that is BSD licensed. This is what Apple used.

But there's a /usr/lib/libreadline.dylib library available! Well, this is just a symlink to /usr/lib/libedit.2.dylib. Editline doesn't have every single feature that readline does, but it's complete enough that, like me, most will just never notice.

Third: Functions

In short: functions allow you to add new functionality to bash by creating functions that can accept parameters and be called by name.

Why: to reduce commonly used sequences of commands to a single name.

How: Define a function. It's as simple as:

```
function func_name { action; action; action; }
```

Add functions to your ~/.bash_profile so they're available for new shell invocations.

Explanation: These functions are the same functions used in shell scripts. They allow a logical grouping of commands—a subroutine, effectively. Functions can even accept positional parameters. The **alias** keyword, often used for a similar purpose, is considered deprecated. Functions can provide everything that aliases do and more, and don't suffer some of the issues that aliases do.

Examples: How often do you create a directory and then change into it immediately? Instead of two steps, make it one step:

```
function mdc { mkdir $1 && cd $1; }
```

Once defined, typing `mdc test` will create a directory named "test" and change into it. You can certainly define multi-line functions. In fact, this is exactly how the bash_completion described above is implemented.

Take a fresh look at
Freeway 5.5
Web design for Mac OS X

With over 70 new and improved features Freeway 5.5 is our best appplication yet!

- Add Image Galleries and Slideshows
- Easily create reliable CSS-Layouts
- Design web sites for the Desktop, iPhone or iPad
- Integrated Site Searches
- Create HTML Email Designs
- Secure FTP
- Make Money with Amazon Affiliates

Freeway is true Web design for the Mac; you design, Freeway codes.

Try Freeway 5.5 today FREE for 30 days only at www.softpress.co

/n software

www.nsoftware.com >

The Best-Selling Internet Communications, Security, and E-Business Components, Now Available On:

Mac OS X and iPhone!

COCOA FRAMEWORKS FOR INTERNET COMMUNICATIONS

IP*Works! eliminates the complexity of Internet development providing easy-to-use, programmable components that facilitate tasks such as sending E-mail, transferring files, managing networks, browsing the web, consuming Web Services, etc.

Internet Communications

- **IP*Works! - [Core Framework]**

A comprehensive framework for Internet development. The core building block for most /n software products.

Components - *IPMonitor, MX, REST, NetCode, RSS, NNTP, SMPP, POP, Rexec, Rshell, Syslog, SMTP, WebDav, SOAP, XMPP, Telnet, Ping, TFTP, FilerMailer, UDPPort, HTMLMailer, WebForm, NetClock, WebUpload, RCP, Whois, FTP, XMLp, HTTP, SNPP, IMAP, MIME, IPInfo, IPDaemon, IPPort, NetDial, LDAP, ICMPPort, MCast, TraceRoute*

Network Management

- **IP*Works! Secure SNMP**

A comprehensive toolkit for building Secure SNMP-based agent and manager applications including advanced SNMPv3 security features, trap handling, and ASN-1 MIB compilation.

Components - *SNMPAgent, SNMPMgr, SNMPTrapMgr, MibBrowser*

File & Streaming Compression

- **IP*Works! Zip**

Suite of easy-to-use, fast, effective components for compression and decompression with advanced features including self-extracting archives, industrial strength AES encryption, and Zip64 archives.

Components - *GZip, Tar, Jar, Zip, ZipStream*

Secure Connectivity & E-mail Confidentiality

- **IP*Works! SSL**

SSL-enabled versions of the components in the core IP*Works! package.

Components - *WebFormS, LDAPS, WebUploadS, RESTS, NNTPS, IPDaemonS, POPS, FileMailerS, SOAPS, HTMLMailerS, TelnetS, HTTPS, FTPS, IMAPS, SMTPS, IPPortS, CertMgr, RSSS, SMPPS, WebDavS, XMPPS*

- **IP*Works! S/MIME**

Components for E-mail and file confidentiality, authentication, and non-repudiation through encryption and digital signatures. Implements the S/MIME standard for digital security.

Components - *CertMgr, SNNTP, SPOP, SIMAP, SSMTP, SMIME, SFileMailer*

- **IP*Works! SSH**

Secure Shell (SSH) enabled client communications components supporting strong encryption and advanced cryptography. A highly-evolved code base, enhanced with enterprise features like IPv6 addressing and 64-bit support.

Components - *SFTP, SShell, SExec, SSHClient, SSHTunnel,* CertMgr

COCOA FRAMEWORKS FOR INTERNET BUSINESS

Built using the same technologies as our award-winning IP*Works! product line, these packages offer native components for Credit Card Processing, Online Payments, E-Banking, Shipping & Tracking, and more!

- **QuickBooks Integrator**
- **E-Payment Integrator**
- **E-Banking Integrator**
- **TSYS Integrator**
- **Paymentech Integrator**
- **FDMS Integrator**
- **FedEx Integrator**
- **USP Integrator**
- **USPS Integrator**
- **PayPal Integrator**
- **Amazon Integrator**
- **SharePoint Integrator**

GET THEM ALL THROUGH OUR COMPREHENSIVE RED CARPET SUBSCRIPTION, $1,499 PER DEVELOPER, ROYALTY-FREE. 20% DISCOUNT FOR MACTECH SUBSCRIBERS.

Fourth: Using bash Special Characters

In short: bash honors several characters as special shortcuts. Make use of them to:

- Type less
- Speed your work
- Make a script generic

Tilde (~)

The tilde character represents the current user's home directory. When, logged in as myself, I type `ls -l ~/Sites`, I'm shown the listing of the "Sites" directory in my home. If you do the same while logged with your user id, you'll be shown the "Sites" directory in your home.

This certainly makes command-line entries easier. For example, no matter where I am in the directory, typing the following:

```
vi ~/.bash_profile
```

will allow editing of .bash_profile from my home directory.

cd –

The cd (change directory) command understands a hyphen ("-") as change to the previous directory. This is a *huge* time saver. It's an easy way to hop between two directories.

Example: You find yourself in the directory /Library/Preferences/Application Support/foo, but then change to your home directory by typing 'cd'. If you realize you need to get back to the previous 'foo' directory, just type 'cd -'.

Magic Space

The Magic Space itself isn't so much of a special character—it still inserts a space as usual—but causes bash to perform variable expansion on the line. Technically, this is a function of *readline*, a library of functions for input.

How: The readline library picks up its configuration from ~/.inputrc. This file doesn't exist by default in an install of Mac OS X, so you'll likely need to create it. Use your favorite text editor and add the following text to ~/.inputrc:

```
$if Bash
  Space: magic-space
$endif
```

We check for bash as there are other programs that use readline that don't support this feature.

Fifth: Terminal.app

In short: I'm on a Mac! How does Mac OS X itself improve the experience of working in a shell?

Drag-n-Drop

Drag and drop is fully enabled from The Finder.

Example: You have a Finder window open to some path that is a bit buried. You then realize you'd like to open a shell at that path. You can open Terminal.app and type 'cd ', and then drag the proxy icon from the Finder window into the Terminal window. (The proxy-icon is document icon in the title-bar of the document window). Terminal converts that to the properly escaped path.

Terminal.app also supports clipping files, so you can drag and drop text out of a Terminal window.

Paste Escaped

You can copy text and paste it into Terminal.app, sure. What happens, though, when the text has specific meaning to the shell, like a path?

Example: If you copied the text:

/Library/Application Support

and pasted it after a 'cd' command, the shell will try to change into "/Library/Application" due to the space. Terminal.app supports a special paste command, though: Paste Escaped Text. It resides under the Edit menu, just under the standard paste command, but even more useful is the keyboard shortcut: control-command-V. I've actually just gotten in the habit of pasting anything into Terminal.app this way.

Back to our example: when you 'paste escaped text', you'll get the proper:

cd /Library/Application\ Support

...and can then simply press the return key. This works exceptionally well with longer strings of text that have embedded characters that need to be escaped from bash's usual expansion or tokenizing.

Finder Integration/Emulation

In short: The previous two commands can certainly be categorized as "Finder integration," but there are specific commands that complete the experience.

open: The open command opens the "file" supplied, just as if you had double-clicked an object in the Finder. In other words, it uses Launch Services data to decide the proper course of action.

Examples: You're in a directory and want to open the Finder at the same location: `open .`

Open a text file in the current directory in the default text editor: `open example.txt`

Open the same text file specifically using TextWrangler (must be installed, of course): `open -a /Applications/TextWrangler.app example.txt`

Open a web page in the default web browser: `open http://www.mactech.com` (yes, this one is awesome).

pbcopy and pbpaste: You have access to the pasteboard from a shell. This is unbelievably helpful.

Examples: If you have output from a shell script that you want to paste into a GUI application, just put it onto the pasteboard:

```
ls -l | pbcopy
```

Also, you can take data from the pasteboard, make modifications and put it back up on the pasteboard. For example, I sometimes take notes in a web-based application that I later need to summarize. I always preface the lines I want to summarize with a hyphen. I can easily create my summary by:

- Select all in the web text area

audioengine **upgrade your music!**

GET 10% OFF plus free shipping when you buy online! Use Coupon Code **MTTEN**

Experience our award-winning sound, high-quality materials and truly useful features.

$199 per pair

Audioengine 2 (A2)
Premium Powered Desktop Speakers

Closes the gap between computer speakers and home audio

*"These **are** the best speakers for your desktop, computer, or media player."* – Connect Reviews

Made for iPod

$119 per set

Audioengine W2 (AW2)
Premium Wireless Adapter for iPod

Unwire your iPod

"Super fast setup and the uncompressed sound is pretty remarkable." – Uncrate

Model A5N shown in Solid Carbonized Bamboo $449

starting at $349 per pair

Audioengine 5 (A5)
Premium Powered Bookshelf Speakers

High-quality audio for your Mac or PC

"There are no other speakers in this price range that come close." – Mac Observer

$99 per set

Audioengine W1 (AW1)
Premium Wireless Audio Adapter

Send CD-quality wireless anywhere with or without a computer

"High quality wireless music streaming that's quick and easy to use." – Register Hardware

Works with all your gear • All cables included • 30-day audition • 3 year warranty

Visit our website for more product info, reviews, and awards: **www.audioengineusa.com**

- Change to a terminal and type: `pbpaste | grep "-" | pbcopy`

- Switch back to the web browser and paste in the results.

There are a lot of other uses for this. It takes a bit of getting used to in order to even remember that this capability exists

Conclusion

There are almost uncountable ways to customize and improve your experience with bash—but you have to start somewhere. These 5(-ish) tips should get you moving toward a better experience when you're staring at a text-based interface.

Media of the month: *Prisoner's Dilemma*, by William Poundstone. Admittedly, I *just* started this book, so I can't quite vouch for it yet. It starts off well, though, and seems like the pace is going to remain consistent.

Of course, I can't fail to mention MacTech Conference—if you haven't signed up, there's still time! Check it out at http://www.mactech.com/conference, and we hope to see you in November!

MT

About The Author

Ed Marczak is the Executive Editor for MacTech Magazine, and has written the Mac in the Shell column since 2004.

Most Advanced Developer Database

Add Value to Applications

Valentina DB for Developers

Object Relational Everywhere

Legendary Speed x 100
Columnar Format
Smarter SQL
Encryption
SSL Compression
Bonjour/Zero-Config
PHP Data Objects
and more ...

Network Model	Relational Model
Tables + Links	Tables, Keys by Values
Direct Pointers	PK, FK, Views, Triggers
Navigation API	SQL, Stored Procedures
	Local & Client/Server

✓ Reports ✓ Server ✓ Studio ✓ ADKs

Platforms:	Windows, Linux, Mac OS X, iPhone
Connectivity:	Director, REALbasic, Revolution, C, C++, C#, Obj-C, COM, VB6, VB.NET, PHP, Ruby, ODBC
License:	**Royalty free** for both local and client/server*

www.valentina-db.com

MacResource Computers & Service

Your Resource for All things Mac

Parts, Parts, Parts

We have parts for iMac, G4/G5 towers, LCD panels G4/G5/Intel iMacs, MacBooks/Pros and Displays.

Logic Boards
G5/Intel iMac: from $179/$399
G5/Intel Towers: $299/$699 & up
G4/G5/Intel Xserve: $149/$299/$899
MacBooks/Pros Call for Pricing

Power Supplies
G4 iMac 15/17/20" $49/$79/$99
G5 iMac 17/20": $99/$129
G5 Towers: $169/$199
G4/G5/Intel Xserves $169/$219/$299
Xserve Raids $239.99

Xserve Processors
G4 1.33GHz DP: $99
G5 2.0/2.3GHz: $149/499
Intel 2.0GHz C2D $699

Logic BD/PS require exchange

AirPort Cards
Standard/Extreme: $69.99

Processors
For, MacPros, G5s, Xserves
Dual Processors (per module):
1.8/2.0/2.3GHz: $199/249/349
2.5DP/QP/2.7DP: $299/599
G5 1.6/1.8GHz: $199/299

Tower Systems
G5 1.6/1.8GHz $429/479
G5 1.8/2.0GHz DP $549/599
G5 2.3/2.5/2.7GHz DP $699/799/899
G5 2.5GHz Quad Dual DVI $999
Intel 2.66/3.06 GHz, MacPro $1499/1799

Intel iMacs!!!
1.83/2.0 GHZ 17" $549/599
2.0/2.16 GHZ 20" $629/649
2.16/2.33 GHZ 24" $729/749

Aluminum iMacs
2.0/2.4/2.66 GHZ 20" Snw Lprd $799/$899
This units are in Factory Refurb Boxes but Have minor scratches on the cases & 90 day warranty.

Need MacBooks/MacBook Pros?
MacBook2.0/2.16 C2D 13" $649/699
MacBook Pro 1.83/2.0 CD 15" $599/799
MacBook Pro 2.4/2.5 C2D 15" $849/899
MacBook Pro2.1/2.4 C2D 17" $949/1199

1-888-Mac-Resource

www.mac-resource.com

We Have G4/G5 & Intel Xserves & RAIDs Even if Apple Doesn't!!!!
G4 Xserve Full Unit: $299
G5 Xserve Cluster Node: $499
G5 Xserve Full Unit: $999
Intel Xserve 2.0ghz Full Unit: $1699
1TB Xserve RAID from $1999
2.8TB Xserve RAID from $2499
5.6TB Xserve RAID from $3699
3.5/7.0TB Xserve RAID: $2899/4299
Blank SATA/PATA Module $199
400GB SATA/PATA Module $399/$349

We carry Fibre Cards. Procs, PSs

Overnight Service Available

Refurbished Displays

Aluminum
20/23 Cinema(DVI): $299/599
30 Cinema(DVI): $999

Crystal
22/23" Cinema(ADC): $249/299
15" Studio LCD(ADC): $49
17" Studio LCD(ADC): $99
17" CRT, ADC/VGA: $49.99

Mac Minis GALORE, GREAT WORKSTATIONS!
1.42G4 GHz/256MB/40GB/Combo: $299
1.66/1.83/2.0 GHZ Intel Minis $349-$449

All Products are refurbished or demo call for more information.

CAMTASIA
[PRONOUNCED cam•TAY•zha]

INTRODUCING
THE ALL-IN-ONE
SCREEN
RECORDER
DESIGNED
FOR MAC

Record screen activity
Edit your content
Add effects
Share your video

Download the free trial and check out our introductory pricing today at: **www.camtasiamac.com**

Camtasia:mac
TechSmith

The Knowledge BIAS

Techniques for keeping things in perspective

by Nathan Toups

Preface

"Becoming an expert in something means that we become more and more fascinated by nuance and complexity. That's when the Curse of Knowledge kicks in, and we start to forget what it's like not to know what we know."

- Chip and Dan Heath, *Made to Stick*

This article is written specifically for IT professionals who struggle with clearly communicating with clients and staff. Because of our high level of knowledge, we have a very special set of challenges facing the relationship between IT and our end-users. I will cover the pitfalls of what I call the knowledge bias and explore two techniques that have worked for me in keeping things in perspective (and memorable) with clients and staff. Though this article is written specifically for IT professionals, the ideas contained here should, with minimum effort, translate to other industries where highly trained experts are expected to communicate with non-experts on a regular basis.

Thesis

Over time, we all have the tendency to become biased in our thinking. This tendency is completely natural. As our learning in a given area accelerates, we become comfortable in our increasingly complex and specialized world and we forget what it is like to live without this knowledge. While continual learning is essential to becoming an expert, these very actions can alienate our clients, cause unnecessary complexity for end-users, and even put us on a path that prevents us from appreciating innovation in our industry.

How do we keep these tendencies in check? We need a foundation and a system. The first step is to clearly identify what personal behavior could cause these inefficiencies. This will give us the foundation. The second step is to create systems that address the pitfalls of communicating clearly so that IT policy is put into perspective for the end-user. I consider this methodology a holistic approach to communicating information technology policy.

So what techniques work for me? Well, that comes in two parts, and I feel they both compliment one another: *shoshin,* the foundation, and "sticky" ideas, the system.

The first technique is Zen Buddhist concept of *shoshin*, which translates to "beginner's mind." In the book *Zen Mind, Beginner's Mind*, Shunryu Suzuki writes, "In the beginner's mind there are many possibilities, in the expert's mind there are few." I have found it critical to my consulting practice to periodically reassess my thinking in the light of shoshin. This helps me keep the "newness" and "joy" of using Mac OS X in perspective. It also allows me to break out of my "expert mind" and truly empathize with my clients.

The second technique is a framework developed by the Heath brothers. Chip and Dan Heath's 2007 business- and marketing-focused, *Made to Stick*, addresses why some ideas "stick" and others do not. A "sticky" idea or concept is one that is memorable enough for non-experts to apply and pass on to others. Imagine changing IT policy in a way that, once an idea is presented, it is passed on with a mind of its own through the organization. This is the power of "sticky" ideas. The book outlines the six characteristics of a "sticky" idea. We will go over these basic concepts and apply them to an example situation.

Combining the powerful Zen Buddhist concept of *shoshin* with the techniques of modern business marketing can help us become mindful experts in our field. So, before we get into the bulk of this article on "sticky" ideas, let us cover the foundation that it is built upon: *shoshin*.

The Foundation: Beginner's Mind

Shoshin, or beginner's mind, is the practice of approaching things without prior judgment and asking questions as if you were a complete beginner. If my eventual goal is to have ideas "stick" with clients and staff, I must first

For iPhone 2G/3G/3GS/iPod
Direct plug-in / no cable.
Includes 2G / 3G support brace.
RS001 ~~$69.95~~ $39.95

For iPhone/iPod with Cable
Cable connection only.
Perfect for iPod touch.
RS008 ~~$69.95~~ $39.95

For All BlackBerry/Smartphones
Cable or direct plug-in.
Works with all USB-port phones.
RS007 ~~$69.95~~ $39.95

10 Reasons RichardSolo 1800 is the best backup battery for iPhone/ Smartphones — and *MacTech* readers save $30. You pay only $39.95

1. It is the only one that actually "latches" onto the iPhone — very stable.
2. Includes free, slim, protective hard case for iPhone 3G/3GS ($24.95 value) that works perfectly with included support brace.
3. Unlike "slipcase" configurations, there is no rear blockage of your cell phone antenna.
4. Licensed and certified by Apple for iPhone 2G/3G/3GS/iPod.

Made for iPod | Works with iPhone

5. Built-in flashlight is surprisingly useful and bright; laser pointer included.
6. Lightweight — you can easily carry it in your pocket, and top up your iPhone and iPod as needed.
7. Choose from two models: direct plug-in or cable. Both are 1800 mAh lithium-ion rechargeable!
8. We support you. Quick email response by the best customer service in the industry, and a full 1-year warranty.
9. **Free bonus** car charger and wall charger included. Charge battery and iPhone in tandem!
10. *MacTech* special price; enter the discount code ***MacTech*** at checkout. Instantly save $30!

Online ordering and blog reviews: www.RichardSolo.com

Enter the discount code *MacTech* at checkout to instantly save $30. You pay only $39.95

Offer cannot be combined with any other discounts or promotions.

Weekly GREAT DEAL — Sign up today at RichardSolo.com to receive our weekly great deal email offer!

Order now online: www.RichardSolo.com

Richard Solo®

iPod is a trademark of Apple Inc., registered in the U.S. and other countries. iPhone is a trademark of Apple Inc. BlackBerry® is a registered trademark of Research In Motion Ltd. Free items require purchase.

be able to have them "stick" with my beginner's mind. Thus, practicing beginners mind allows me an opportunity to break out of expert thinking and gain unique insight that would otherwise be impossible.

In Blanch Harman's lecture on beginner's mind, she tells a story of a child named Indigo, who picks up a small object, studies it, hangs it against a table, puts it in his mouth, and otherwise examines it. Then she goes on to explain that you or I would call this object a spoon, and that we all know that it is used for soup. Her point is that Indigo personifies the innocence of asking as a true beginner "What is it?"

This story may not sound like much, but the core message is important: approaching established, even seemingly boring, things with a beginner's mind can open a world of possibilities and give us unique insight into how others may first experience the things around them. Imagine the joy (and chaotic power) of a child discovering that a spoon makes a wonderful catapult.

Shoshin in Practice

In practice, it is the very question, "What is it?" asked with a clear mind, that is so important. Ask yourself this question the next time you run into a problem you cannot solve with your expert mind. It is quite liberating. When I ask myself this question, I am giving myself permission to not be an expert for a moment. I attempt to look at an error message, policy loophole, or client question as if I, myself, was the beginner. I give myself permission to install a clean copy of Snow Leopard and just play. I give myself permission to explore a new social media app for the iOS 4 without prejudice. Or, I simply browse Apple's website as if I had just considered buying an Apple product for the first time.

Many of my clients tell me that the biggest differences between my consulting practice and those they have used before are my willingness to listen, my approachability, and my patience. I credit this to the value I place on *shoshin* for problem solving and self-discovery.

Shoshin is a mindset more than a system. Although it is a very powerful tool for the individual, by itself, it does not explain how to clearly communicate with the non-expert. So now that we have laid the foundation for thinking as a non-expert, we need a system for creating "sticky" ideas for the non-expert.

The System: Making it Stick

> "Lots of research in economics and psychology shows that when we know something, it becomes hard for us to imagine not knowing it. As a result, we become lousy communicators. Think of a lawyer who can't give you a straight, comprehensible answer to a legal question. His vast knowledge and experience renders him unable to fathom how little you know. So when he talks to you, he talks in abstractions that you can't follow. [And] we're all like the lawyer in our own domain of expertise."
>
> – Chip and Dan Heath in an interview with Guy Kawasaki

We have established that the curse of knowledge is seemingly inescapable. As we become experts, and we start

SmallTree

The Mac Networking Experts

Manage your workflow better with Shared Storage that everyone can afford

Check out all our Ethernet Shared Storage Solutions at

GraniteSTOR.com

Winner of the 2010 Vidy Award

www.small-tree.com • 1.866.782.4622 • 866.STC.4MAC
7300 Hudson Blvd., Suite 165, Oakdale, MN 55128
www.GraniteSTOR.com

The Mac Networking
1.866.782.4622 • 866
Home Downloads Contact Us
Simple, Affordable, Effective shared storage solutions for everyone
GraniteSTOR
ST-RAID Final Cut Pro & Video Solutions
abcSAN ProTools & Audio Solutions
What is ST-RAID
What is abcSAN

casper SUITE

HELPING THE ENTERPRISE SUCCEED WITH THE APPLE PLATFORM

98.21%

customer retention over a 7 year period

500,000+

seats under management

700+

commercial, education and government customers

Complete product information { jamfsoftware.com

JAMF software

generating unique perspectives and ideas in our field, it becomes more difficult for us to relate these ideas to non-experts. This is not to say that it is impossible to communicate to non-experts; it is merely difficult to do so without a system to help ideas "stick" with end-users.

What Is a Sticky Idea

A "sticky" idea or concept is one that is not only understood by almost anyone, but one that resonates enough with a non-expert that the person feels comfortable spreading it. This is not magic. It happens every day. The best urban legends are very sticky. Diet fads can be sticky. Apple, Inc. is the master of sticky ideas. Each of Apple's ad campaigns begs to be repeated (particularly by the non-expert).

So some ideas are naturally sticky, like an urban legend. Other ideas are constructed, like an Apple ad campaign. For us, creating a sticky idea helps us to connect with our end-users. It prevents us from alienating them with command line references, acronym overload, and unnecessary tech specs. It allows us to cut to the core concept, promotes clarity of thought, and helps us clearly define the scope of a project or policy. Luckily, the Heath Brothers outline the six characteristics of a "sticky" idea in their book, *Made to Stick*, in the form of an acronym: SUCCESs.

The six characteristics are as follows:

Simple – Find the core of an idea
Unexpected – Grab people's attention by surprising them
Concrete – Make sure an idea can be grasped and remembered later
Credibility – Give the idea believability
Emotion – Help people see the importance of an idea
Stories – Empower people to use an idea through narrative

Now, a "sticky" idea does not have to include all six characteristics, but as the Heath brothers say, "the more the merrier." So, let us explore each characteristic of the SUCCESs model for sticky ideas.

Because I like reading examples that apply to the real world, we are going to create a scenario for which a "sticky" idea is critical to transmit a concept to the end-user. We will use one of my favorites: Backups and Data Security.

The Scenario

Say your client has a small office with a mix of desktops computers, laptops, and mobile devices. It is also a mixed environment of Windows and Mac OS X. They have some backups in place on some computers, but they have no official policy. iOS 4 devices may or may not be password protected and do not have remote wipe set up. Despite the lack of a policy, protecting the office data is very important to your client. If any of the mobile devices fell into the wrong hands, it could spell disaster for the company.

Drafting the Message:

Without going into details, we know the office needs two things:
- A comprehensive security policy that is measurable and enforceable
- A complete backup system that is easy to use, preferably automated, and testable.

So What is the Message?

Using my "expert" mind, I have a tendency to explain the project goal as "Having a protected data workflow policy." It sounds impressive: it is broad enough to cover my bases for data security, device security, data redundancy and data backup versioning. However, it does not consider *shoshin*—or how the uninitiated mind will react—and it sorely lacks in stickiness. Can you imagine an intern repeating this goal to explain how the company deals with security?

Using the "sticky" guide, I might explain the same project goal as, "All data protected, all devices secure." Do you feel difference? Now imagine that same intern repeating this to explain how the company deals with security. Lets break down the second project goal using the SUCCESs framework for: "All data protected, all devices secure."

Simple

The idea is not complex. We are talking about two things: data and devices. This is important so that end-users can grasp on the core idea quickly.

Unexpected

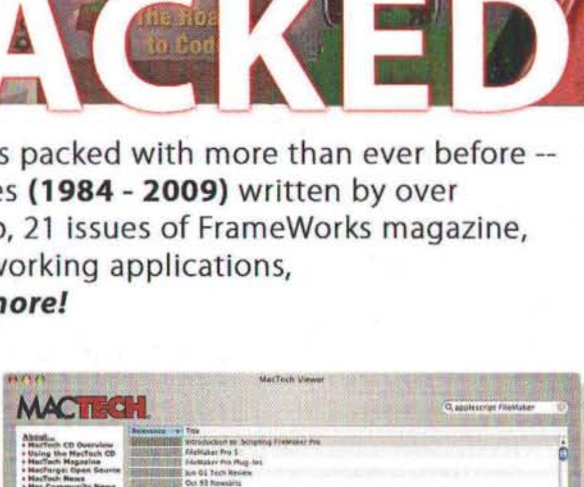
Unexpected ideas are valuable because they are inherently memorable. The boldness of the statement above may or may not work to catch all end-users by surprise, but if this office has always acted fuzzy on how backup policy and security policy work, a boldly stated yet elegantly simple goal may be quite unexpected. This is especially true if previous IT policy was wishy-washy in the past.

Concrete

We have very concrete terms for this goal. All of the data is protected and all of the devices are secure. There is no guessing what data should be protected or which devices might need to be secured. Concrete ideas are easier for end-users to grasp than abstract conceptual thinking. Let us say an end-user needs to figure out if syncing her office contacts to her personal phone is acceptable. She can ask herself, "Is the data protected and is the device secured?" If the answer is "No" or "I don't know," she knows it needs to be checked out by IT.

The MacTech DVD - Volumes 1.01-25.12 is packed with more than ever before -- over 3200 articles from more than 290 issues **(1984 - 2009)** written by over 870 experts, all 29 issues of Apple's develop, 21 issues of FrameWorks magazine, 100+ MB of source code, MacTech Viewer, working applications, full documentation, demos for techs, ***and more!***

Everything is displayed in the very fast, very searchable ***MacTech Viewer!*** An application that has been designed specifically with Techs in mind. Search quickly through 25 years of great information provided by MacTech. Information to save you time, and make your life easier.



Volume Number: 22
Issue Number: 4
Column Tag: Programming

AppleScript Essentials

Introduction to Scripting FileMaker Pro

by Benjamin S. Waldie

For the past couple of columns, we have discussed various ways to store and access data using AppleScript. One column provided an introduction to Database Events, a background application in Mac OS X 10.4 and higher, which allows AppleScripts to interact directly with SQLite databases for the purposes of storing and accessing data. Another column explored methods of storing and accessing data in script properties and property list files. This month, we are going to continue discussing data storage and access, this time, using FileMaker Pro, a third-party commercial database application.

For the purposes of following along with this month's column, if you do not own FileMaker Pro, you can download a fully functional 30-day trial from **http://www.filemaker.com/**

All AppleScript code covered in this month's column was written and tested with FileMaker Pro version 8.0.1. Therefore, some of the AppleScript terminology discussed, may not function with

Mac

Universal

Requires Mac OS X v. 10.4.5 or later

Toll Free 877-MACTECH, Outside US/Canada: 805-494-9797 • http://www.mactech.com/dvd/

Credible

Hopefully this one is a non-issue. You are the IT expert. They are paying you money to solve problems. You have established credibility.

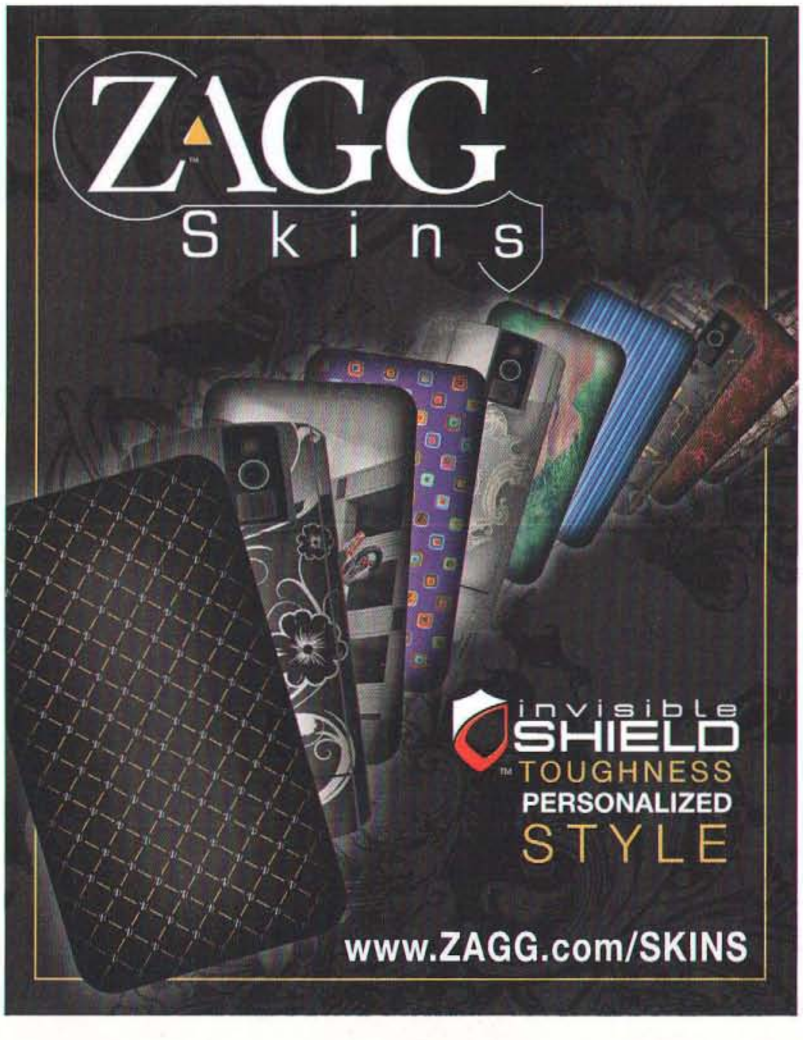
Emotion

Though this statement is not the peak of emotion, it is significantly more emotional than, "A protected data workflow policy." "All data protected, all devices secure" includes all end-users. There is no question if their devices or data will be included. There is an emotionally driven certainty that things are going to be taken care of.

Story

The final item, story, is much easier to draft with our latter example. For instance, it would be easy to create an analogy to a chain in which security and backups are only as strong as their weakest link, therefore to gain maximum strength out of both, all data and all devices must be dealt with to avoid a weak link.

As you may have noticed, I have not covered how I am going to pull this off. For the scope of this article, it does not actually matter. A "sticky" idea provides a firm foundation to build that policy on, and the details are not the concern of the average end user. Overall, our job behind the scenes really does not change from the verbose technical explanation to the "sticky" one. We would be able to implement the same backup and security policies for either goal. The fundamental difference is that the second goal allows us to break through our "Curse of Knowledge" by using *shoshin* to keep it relatable and then crafting memorable and repeatable ideas that end-users can understand.



Conclusion

If you are like me, you are a geek from the moment you wake up to the moment you fall asleep at night (and sometimes geeky dreams even creep in there as well). I work on OS X and Linux servers all day, and then when I get home, I work on my personal OS X and Linux servers for fun. We live in a unique reality distortion field in which our work and play are, many times, a glorious blur. This is a great place to be, but we must be careful to think about those around us who do not share this interest. We must keep the non-expert-end-user in mind whenever we make decisions that may affect them. We need to consistently evaluate our attitude to the technology around us, with a focus on discovery and growth.

I hope that I have made a convincing argument for the value of incorporating *shoshin* and "sticky" ideas into your sysadmin tool belt. To be honest, I would not be the least bit surprised if some of you are already using some or all of the techniques presented above.

If these ideas resonated with you, I highly recommend diving in deeper by reading *Zen Mind, Beginner's Mind* and *Made to Stick*.

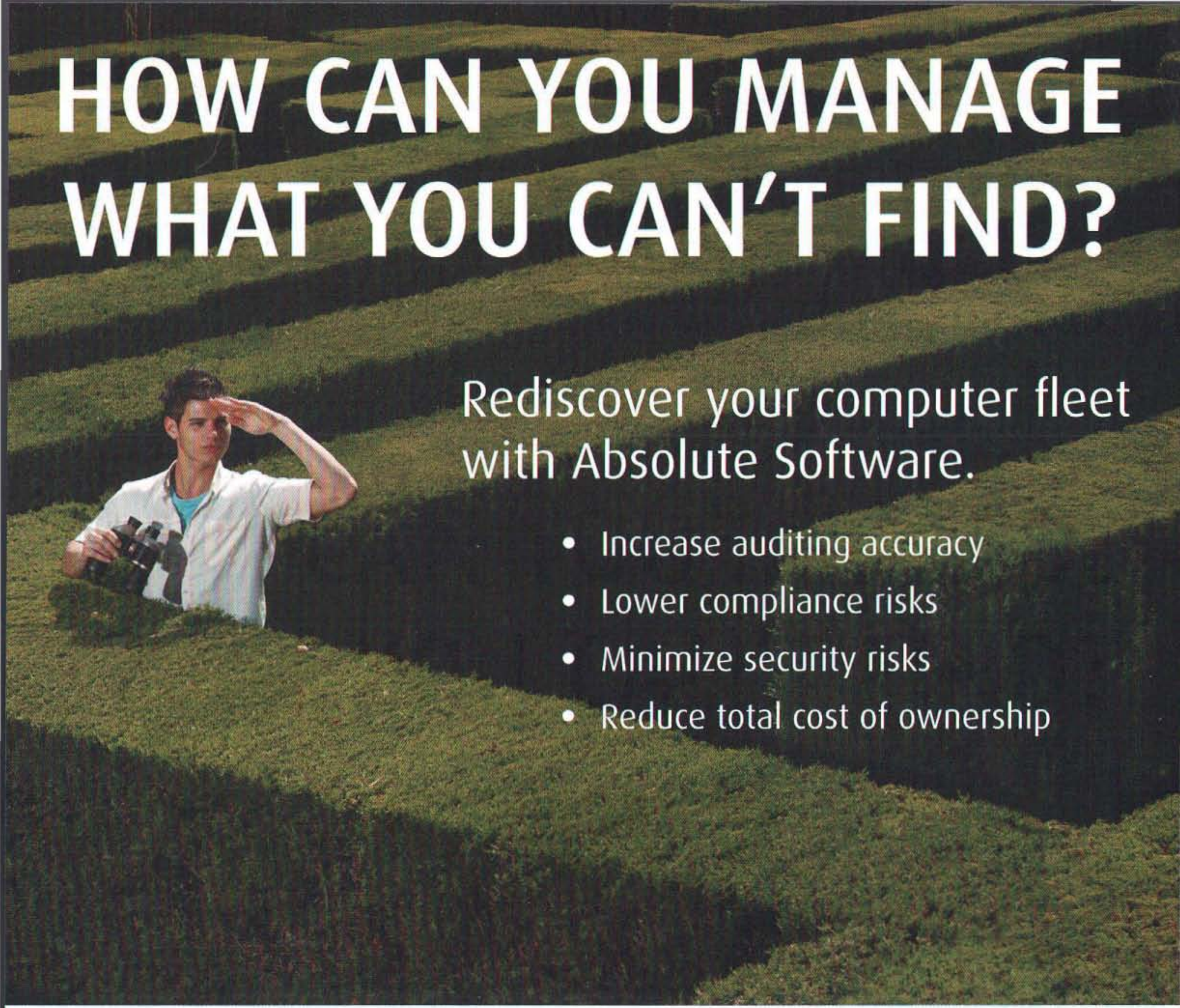
Bibliography

Hartman, Blanch. "Beginner's Mind." http://www.intrex.net/chzg/hartman4.htm. 2001.
Heath, Chip, and Dan Heath. *Made to Stick: Why Some Ideas Survive and Others Die*. 1st ed. Random House, 2007.
Kawasaki, Guy. "The Stickiness Aptitude Test (SAT) and Ten Questions with Chip and Dan Heath." *How to Change the World*. http://blog.guykawasaki.com/2007/01/the_stickiness_.html. January 09, 2007.
Suzuki, Shunryu. *Zen Mind, Beginner's Mind*. 1st ed, Shambhala, 2006.

MT

About The Author

Nathan Toups lives in Austin, Texas with his lovely wife and his lazy (but loveable) cat. When he isn't working with client/collaborators through his consulting firm, rojoroboto, he enjoys participating in a myriad of personal and community focused activities to improve his well-being and the well-being of others. You can contact him at nathantoups@rojoroboto.com


HOW CAN YOU MANAGE
WHAT YOU CAN'T FIND?
Rediscover your computer fleet with Absolute Software.
• Increase auditing accuracy
• Lower compliance risks
• Minimize security risks
• Reduce total cost of ownership

Find and manage your Mac and PC computers with Absolute® Software. From the largest corporations to your home office, our Computrace®, Absolute Manage and LoJack® for Laptops solutions help you improve data protection, simplify computer lifecycle management and recover stolen computers.

For a FREE demo of Absolute Manage visit:

www.absolute.com/rediscover

Absolute®Software

The absolute best way to track, manage & protect your digital world.

From iPhone to iPad

Converting an iPhone app to a Universal app

by Rich Warren

Getting Started

So, you've made an iPhone app. Customers are buying it. They're happy. You're happy. The world is filled with sunshine and flowers…but there is one thundercloud on the horizon—the iPad.

Recently, you've noticed a number of requests for an iPad version in your online support forums. Of course, given the state of the Internet these days, it was probably a flood of "IPAD EPIC FAIL" messages whining about how crappy (their word, not mine) your app looks in x2 mode. Clearly, something needs to be done, but what?

Examining the iPad Alternatives

You have two basic choices for providing iPad support. You can either build an entirely new application for the iPad (the infamous, more-expensive "HD" version), or you could build a universal application that runs on both devices. Both approaches have their strengths and weaknesses.

On the surface, the most obvious consideration appears to be cold, hard cash. Admit it, we all want more money (and that's not necessarily a bad thing). The math seems simple. If we provide a Universal application, all our iPhone users will get their iPad copy for free. If we create a second app, we can force them to pay for both devices.

But, let's think about this a bit. You may sell a number of HD versions to existing customers, but you're also going to upset a lot of people. Most of us resent paying multiple times for the same product. Unless the iPad version is considerably different from the iPhone (vastly improved artwork, better UI, whatever), you should expect a certain amount of blowback.

On the other hand, a Universal app can become a powerful marketing tool. After all, a potential customer is much more likely to buy a new app if they know it will run on both their iPhone and their iPad. And let's face it. Unless you're already doing unbelievably well, potential new customers will always greatly outnumber your existing customers.

Of course, from a design standpoint, the economics should be a minor factor in this decision. You must also consider how the organization of your code will affect your ability to create, expand and maintain your application.

In many ways, building separate applications may be the simplest approach. It gives you the most flexibility; you decide exactly how much code you will share between the two projects, and how (or weather) you will keep that code in sync. The possibilities range from having two, completely separate code bases to producing two build targets that share a single copy of the code within the same project. You can also leverage a wide range of existing software engineering techniques—Frameworks, shared libraries, version control repositories, branching and merging code, etc.

Building an entirely new application is especially appropriate when the iPad's version will be considerably different from the iPhone's. After all, a mobile app's user interface drives much of the overall design. The iPhone's small screen demands restraint. You can only present a few controls on the screen at any one time. Instead of trying to squeeze in a full range of features, most iPhone apps focus on making the core functionality elegant and simple to use. They also emphasize short, frequent interactions—the type of things users might do while standing in line at the grocery store. You get in, check some messages and get out fast.

The iPad, on the other hand, allows a little more breathing space, both in the number and type of controls that can be placed on screen, and in the expected focus of the application itself. A good application still needs focus, of course, but iPad users can reasonably expect a richer interface and broader range of features. In many cases, an iPad may replace carrying a full laptop. It is not unreasonable for users to spend considerably more time with their apps. While the iPhone may be fine for standing in line, the iPad user expects to get work done. This often means the iPad version will need (or at least should use) a sizeable amount of custom code not included in the iPhone version. In these cases, it may be best to build two separate applications.

A Universal app, on the other hand, will always be more complicated than a device-specific version. You will build a single application that runs on both devices. As a result, your application will contain all the code and resources needed for both versions. You must perform runtime checks to determine when device-specific code should run. The more device-specific code you use, the more complex the design becomes.

Still, for a large number of applications, the conversion from iPhone to iPad is fairly straightforward. Yes, you will want to tweak the interface, but most of the code remains unchanged. For example, many iPhone applications use `UITableViews` and `UINavigationControllers` to navigate through hierarchical information. These can be easily converted to an iPad application by embedding the iPhone's table into a `UISplitViewController`. This provides a richer iPad-specific user interface that takes full advantage of the greater screen size, while requiring only minor changes to the code itself. These applications often make excellent Universal apps.

So, Where Do We Go From Here?

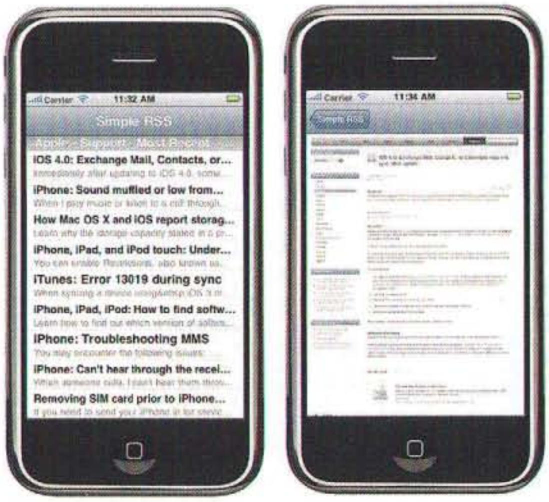
In the rest of this article, we will convert a simple iPhone application into a Universal version. Our starting app is named Simple RSS. When launched, it reads Apple's iPhone Support feed, and displays a list of available articles. When the user taps on an article, it opens the corresponding web page in a `UIWebView`. OK, it's a ridiculous little application, but it will allow us to explore a number of interesting techniques and issues.

Before we go much further, let me state that this article was written based on Xcode 3.2.3, iOS 4.0 for the iPhone and iPhone OS 3.2 for the iPad. Both Xcode and iOS are fast moving targets, so some of the details may change by the time you read this.

Building a Universal App

So, to get started, download the source code from ftp://ftp.mactech.com.

Open the original Simple RSS project, and let's get started. First, build and run the application. Browse through the list of articles. Move back and forth between the list and detail view. Just get a feel for how things work.


Carrier
11:32 AM
Simple RSS
iOS 4.0: Exchange Mail, Contacts, or...
iPhone: Sound muffled or low from...
How Mac OS X and iOS report storag...
iPhone, iPad, and iPod touch: Under...
iTunes: Error 13019 during sync
iPhone, iPad, iPod: How to find softw...
iPhone: Troubleshooting MMS
iPhone: Can't hear through the recei...
Removing SIM card prior to iPhone...
Carrier
11:34 AM
Simple RSS

Simple RSS running on the iPhone

Now take a second to look at the code. There are two root view controllers: `RootViewController` and `DetailViewController`, each with its own NIB. Not surprisingly, the `RootViewController` manages the list of articles, while the `DetailViewController` manages the display of a specific web page.

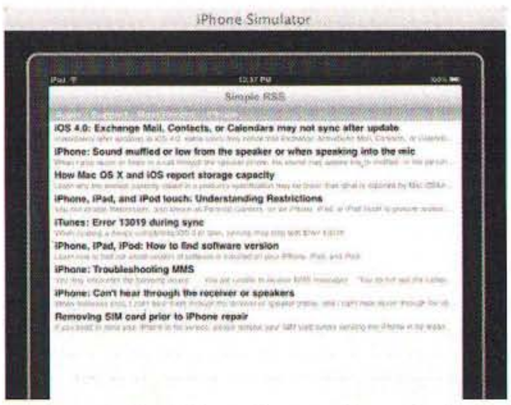
The project also has an app delegate, `SimpleRSSAppDelegate` and three NIB files. `RootViewController.xib`, `DetailView.xib` and `MainView.xib`. The first two correspond with the previously mentioned controller classes. `MainView.xib` acts as the launch point for the iPhone app. It sets up the window, and then loads the root view. Most of our work will focus on these files.

In addition, the project has a few classes that support parsing the RSS feed. `RSSParser` (not surprisingly) parses the feed, creating a list of `Entities`. Each `Entity` represents a single article from the RSS feed (or, at least the parts that we are interested in). We also define a `stack` category on the `NSMutableArray`. This simply adds `push:` and `pop` methods to the existing `NSMutableArray` class; however, these methods are only used inside the `RSSParser`. Feel free to ignore these files, since we will not touch them during the rest of this tutorial.

Getting the Basics Up and Running

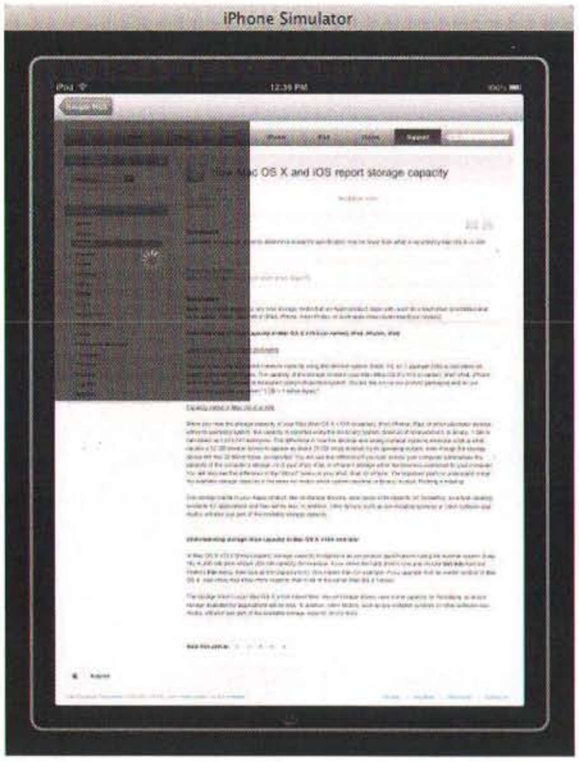
Our first step involves creating a Universal target. Fortunately, Xcode automatically handles most of the work for us. Start by selecting the SimpleRSS target from the Groups & Files view, and then select Project ® Upgrade Current Target for iPad.... Select One Universal application in the popup sheet, and then select OK. Xcode makes the changes needed to run this app on both devices. This includes altering the deployment target, and adding a new iPad specific NIB file: `MainWindow-iPad.xib`.

Let's test it out. Build and Run. It should launch again in the iPhone simulator. The iPhone app should appear and function as it did before. So far, so good. Terminate the app, and then change the Active Executable to SimpleRSS - iPad Simulator 3.2. Build and run again. It should now launch in the iPad simulator.


iPhone Simulator
iPad
Simple RSS
iOS 4.0: Exchange Mail, Contacts, or Calendars may not sync after update
iPhone: Sound muffled or low from the speaker or when speaking into the mic
How Mac OS X and iOS report storage capacity
iPhone, iPad, and iPod touch: Understanding Restrictions
iTunes: Error 13019 during sync
iPhone, iPad, iPod: How to find software version
iPhone: Troubleshooting MMS
iPhone: Can't hear through the receiver or speakers
Removing SIM card prior to iPhone repair

The unmodified Universal app running on the iPad

Well, it's not very pretty. We'll fix that in a bit. For right now, try selecting one of the articles.



The DetailView, however has some problems.

Obviously, there are some problems here. First, `DetailView` uses a shaded overlay to show that the page is loading. This overlay is sized correctly for the iPhone, but does not take into account the iPad's larger screen. However, there is a more serious problem. Once the page loads completely, it crashes, throwing the following exception: `+[UIView animateWithDuration:animations:]: unrecognized selector sent to class 0x217d89c.`

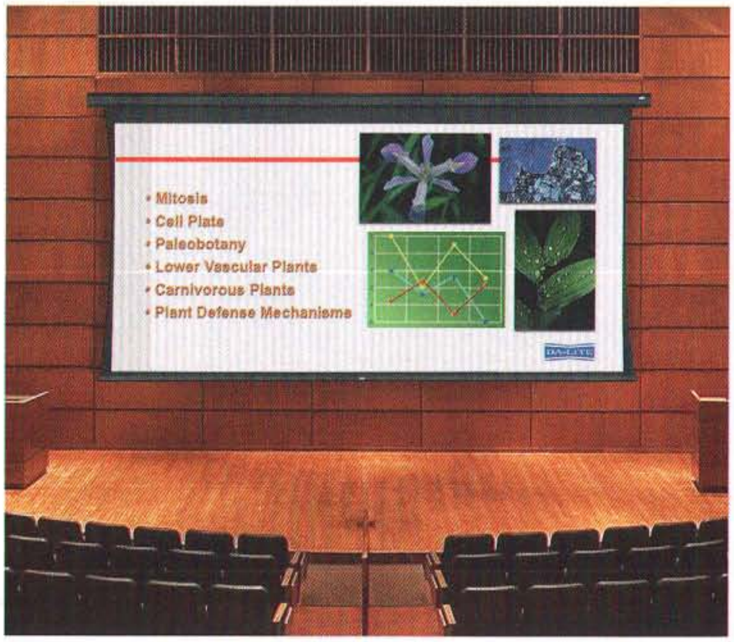
Looking up `+[UIView animateWithDuration:animations:]` in the documentation, you can see that this method is part of the new block-based API added for iOS 4.0. Unfortunately, our iPad still runs iPhone OS 3.2. So, the real question is, why did this compile in the first place?

Let's take a quick detour and look at how Xcode supports older devices. Right click on the **SimpleRSS** target and select **Get Info** to view the compilation options. At the top, you will see that **Base SDK** is set to **iPhone Device 4.0**. Scrolling down, the **iPhone OS Deployment Target** is set to **iPhone OS 3.2**.

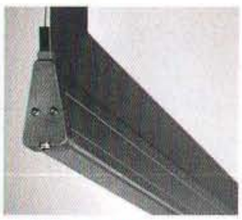
This is Apple's recommended approach. Set the base SDK to the highest option available, then set the deployment target to the earliest device you wish to support. This allows newer devices to take full advantage of the new SDK and runtime, while still providing some backward capability. Xcode manages this by weak linking the new SDK. Everything compiles fine, but if you try to use the new functions, classes or methods, on an earlier version of the OS, the application will crash.

This means you must perform runtime checks to protect older devices from the new SDK. There are a few simple rules. Any unavailable C functions will have `NULL` function pointers, any unavailable class names will return `nil` from `NSClassFromString:`, and any unavailable methods will

The Best Tensioned Screen Just Got Bigger



**Tensioned Cosmopolitan® Electrol®
Now Available Up To Sixteen Feet
SEAMLESS**


Weighted aluminum slat bar.

Two piece aluminum extruded case.

DA-LITE®

For a free catalog and screen recommendations, call toll free or visit us online.

1-800-622-3737

www.da-lite.com • info@da-lite.com
574-267-8101

Register for Early Bird Discounts and SAVE!

Choose from more than 45 classes!

- Beginner to Expert
- Enterprise to ISV to Indie Developer

PLUS! Marketing Track
Increase sales of your app in the app store!

Technical Program Online at iphonedevcon.com

Developing for the iPhone or iPad?

Attend **iPhone/iPad DevCon**2010

San Diego • Sept. 27-29, 2010
www.iphonedevcon.com

A BZ Media Event

return NO from the containing object's `respondsToSelector:` method.

With that in mind, let's take a look at `DetailViewController`'s `webViewDidFinishLoad:` method.

Buggy webViewDidFinishLoad: Method

```
- (void)webViewDidFinishLoad:(UIWebView *)webView {

    [UIView animateWithDuration:1.0 animations:^{
        self.activityHUD.alpha = 0.0;
    }];

}
```

We could just replace this with an iPhone OS 3.2 compatible code, but let's go ahead and add a runtime check. Newer devices will use the block-based code with a 1.0 second fade out, while older devices use the standard 0.2 second fade. When you compare the iPhone and iPad versions, you should be able to see the difference.

Correct webViewDidFinishLoad: method

```
- (void)webViewDidFinishLoad:(UIWebView *)webView {

    // Make sure we only call the block-style animation on
    // supported machines (ios 4.0 or later).

    if ([UIView respondsToSelector:
            @selector(animateWithDuration:animations:)]) {

        // iPhone shade fades out over 1 second

        [UIView animateWithDuration:1.0 animations:^{
            self.activityHUD.alpha = 0.0;
        }];

    }
    else {

        // iPad shade fades out in 0.2 seconds

        [UIView beginAnimations:animationID context:nil];
        self.activityHUD.alpha = 0.0;
        [UIView commitAnimations];

    }
}
```

Here, we make sure `UIView` responds to the `animateWithDuration:animations:` method. If it does, we use the iOS 4.0 block-based animation. If it doesn't, we default back to older animation code. Of course, our legacy code needs a new `animationID` variable. You must define this at the top of the `DetailView.m` (just under the imports).

Just Mobile AluPen™ is a chunky pencil-shaped stylus that gives you precise control over your iPad, iPhone or iPod touch.

Sculpted from aluminum, with a soft rubber nib, the AluPen™ makes drawing and writing a uniquely smooth experience. That is why it's the essential accessory for creative tableteers.

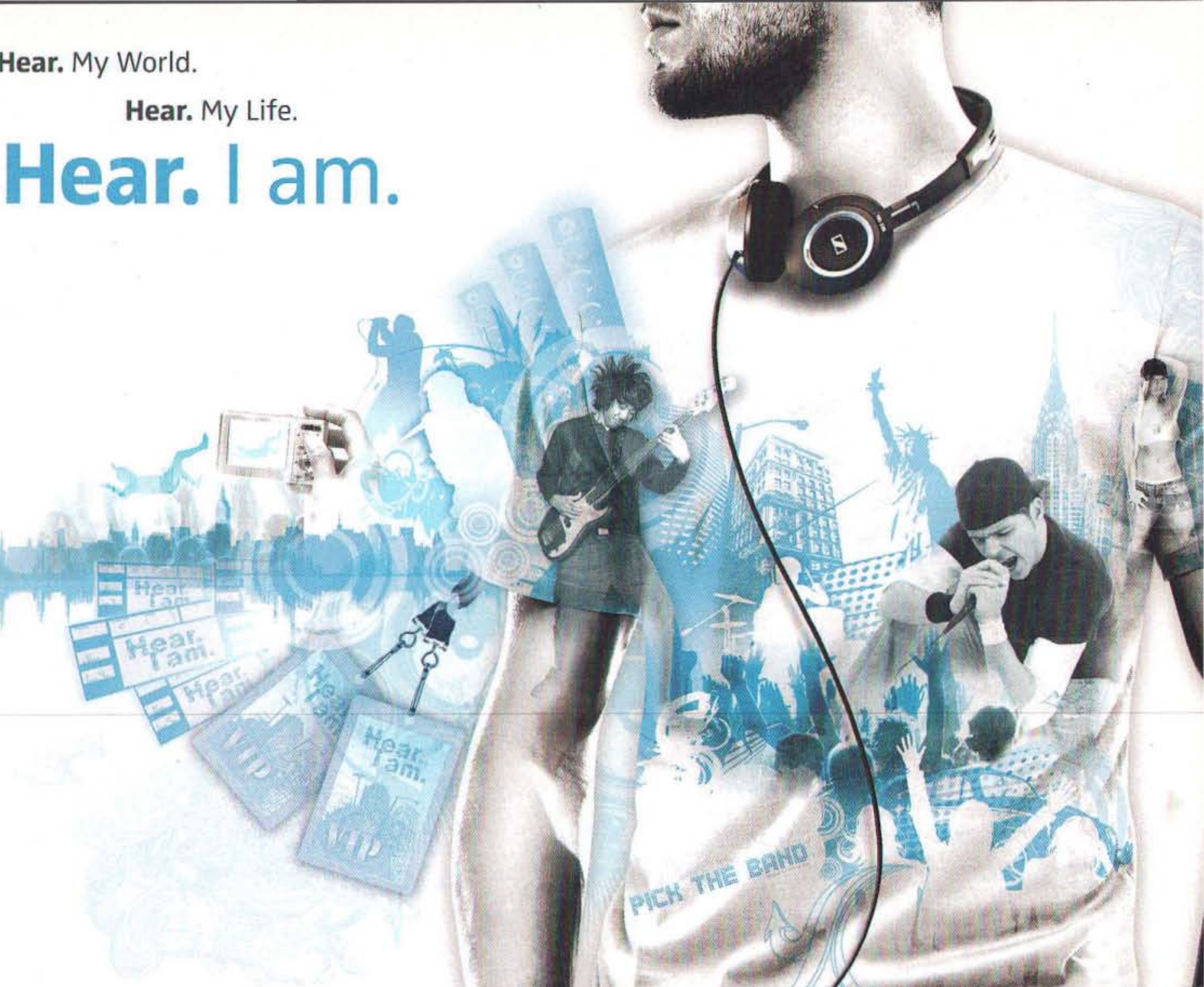
www.just-mobile.com

just | mobile®

Hear. My World.

Hear. My Life.

Hear. I am.



The soundtrack to your life is your own.

Listen, perform or experience it your way with Sennheiser at CMJ 2010

MUSIC MARATHON & FILM FESTIVAL
OCTOBER 19-23 NYC | CMJ.COM/MARATHON

DISCOVER • SHARE • VOTE
YOU COULD **WIN*** AT
Hear I am.com
*NO PURCHASE NECESSARY

Headphones
& Microphones

SENNHEISER

animationID Definition

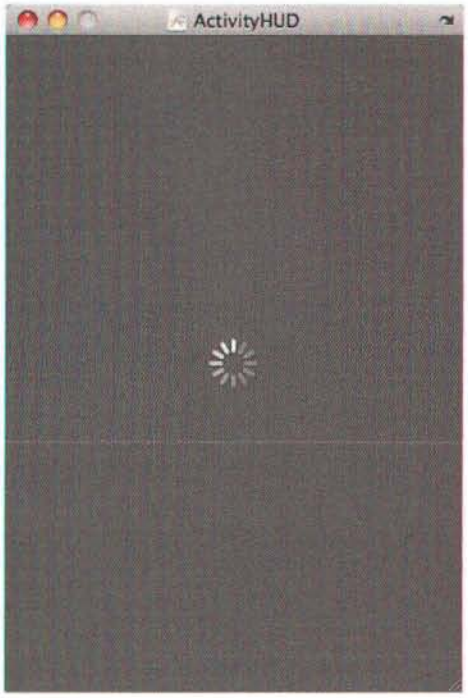
```
// needed for old-style animation
static NSString* animationID = @"HUD fade out";
```

While that fixes the immediate problem, this bug raises an interesting question. When you're writing code to support older deployment targets, how do you know when you're using features from the newer SDK? If you're like me, you lean heavily on Xcode's auto-completion. It is very easy to accidentally add unsupported method calls without ever realizing it.

Obviously, somewhere under the hood the compiler knows what's going on, since it correctly weak-links the newer SDKs. However, I haven't yet found any way to extract this information from Xcode. Instead, the recommended approach is to test your application thoroughly on every supported device.

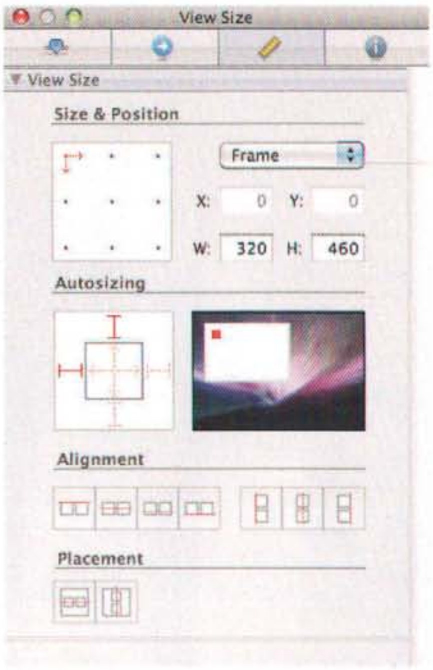
While thorough testing is necessary, I don't think it's an adequate solution for this problem. Unsupported functions, classes and methods could crop up anywhere in the application. Guaranteeing that your tests cover every possible code branch is (except in the most trivial cases) simply not feasible. As a result, little code landmines could lurk in rarely-called branches of code just waiting to crash the app. Hopefully Apple will give us some sort of static analyzer or compiler warning to catch calls to the newer SDKs, but for now, testing is the only option we have. Do the best you can.

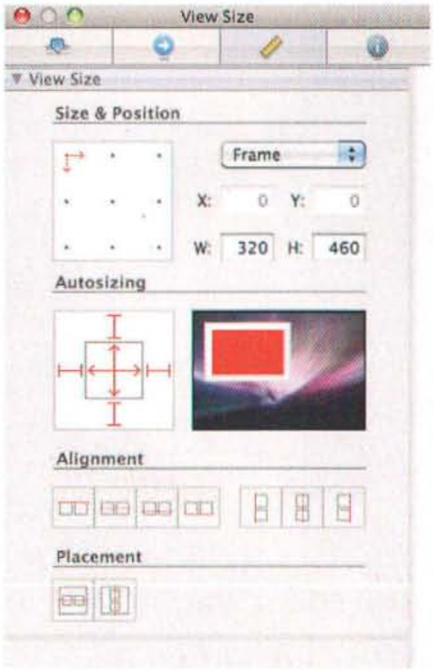
Now, lets fix the `DetailView` shader. Open `DetailView.xib` in Interface Builder, and double click on the ActivityHUD view. This should bring up a gray window with a white activity indicator in the center.



ActivityHUD View

In the Inspector, click the Size tab. Notice that its position is locked to the upper left corner, and that it does not resize to fit. Lock it to all four sides and set it to resize both horizontally and vertically, as shown below.

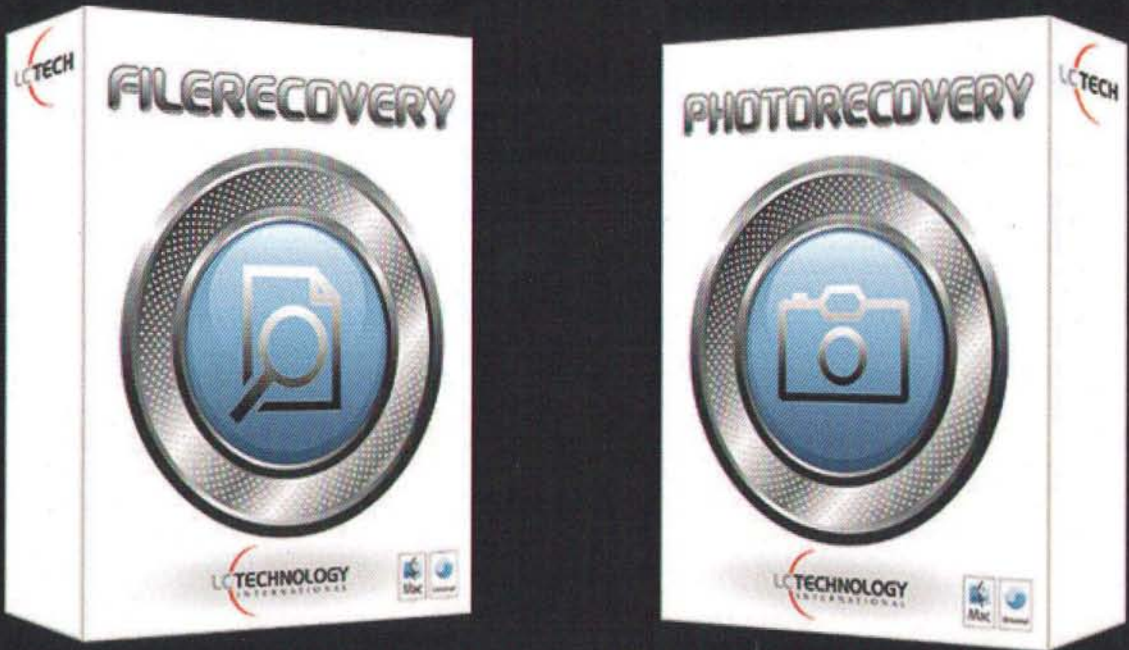




Change the Autosizing settings as shown.

Note: On my laptop the ActivityHUD's display got kind of funky after this change. I had to close the window, then re-open it, and then reposition the activity indicator. Your mileage may vary.

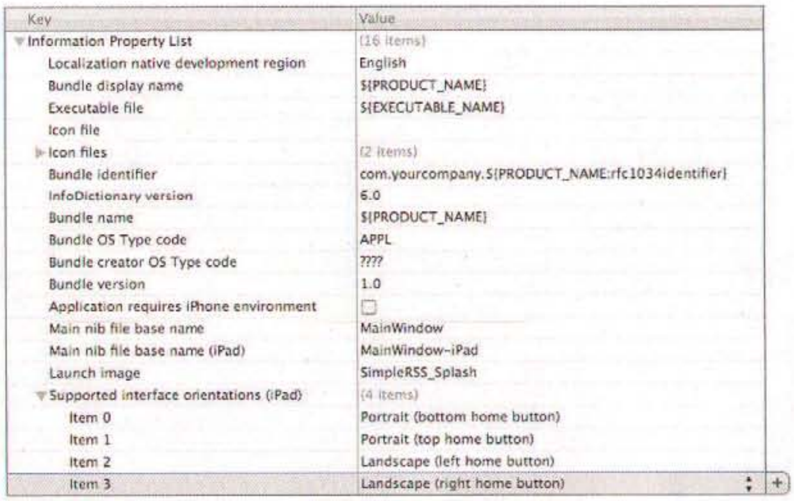
DATA RECOVERY FOR YOUR MAC



WWW. LC-TECH.COM 1-866-603-2195

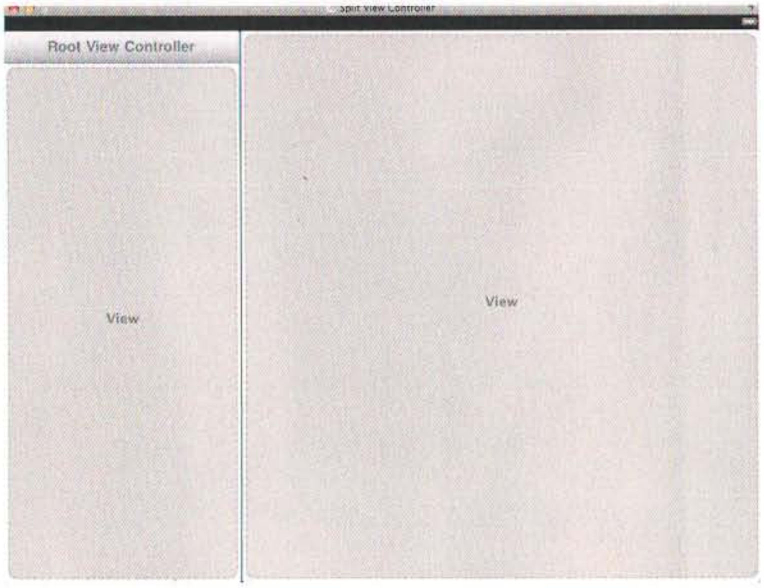
LC TECHNOLOGY INTERNATIONAL

Mac Universal



Key	Value
▼ Information Property List	(16 items)
Localization native development region	English
Bundle display name	${PRODUCT_NAME}
Executable file	${EXECUTABLE_NAME}
Icon file	
▶ Icon files	(2 items)
Bundle identifier	com.yourcompany.${PRODUCT_NAME:rfc1034identifier}
InfoDictionary version	6.0
Bundle name	${PRODUCT_NAME}
Bundle OS Type code	APPL
Bundle creator OS Type code	????
Bundle version	1.0
Application requires iPhone environment	☐
Main nib file base name	MainWindow
Main nib file base name (iPad)	MainWindow-iPad
Launch image	SimpleRSS_Splash
▼ Supported interface orientations (iPad)	(4 items)
Item 0	Portrait (bottom home button)
Item 1	Portrait (top home button)
Item 2	Landscape (left home button)
Item 3	Landscape (right home button)

SimpleRSS-Info.plist with the iPad Launch Orientations Added



Blank Split View

Be sure to save your changes in Interface Builder. Then build and run the application again. The shader is definitely bigger, but it's still not quite the right size. That's easy enough to fix. Open `DetailView`'s `viewDidLoad` method, and modify it as shown below.

DetailView.m's viewDidLoad Method

```
- (void)viewDidLoad {

    self.activityHUD.alpha = 0.0;

    // make it the same size as the web view
    self.activityHUD.frame = self.webView.frame;

    [self.view addSubview:self.activityHUD];

    self.webView.delegate = self;

    [super viewDidLoad];
}
```

Here, we simply set the **activityHUD**'s frame equal to the **webView**'s frame. Build and run it again, the iPad version should work now. Run the iPhone version again to verify that it still works. Excellent. We now have a working Universal app. Of course, the iPad version isn't very iPadish. Don't worry, we're getting there.

Rotating and Launching in any Orientation

Our iPhone application only operates in portrait mode. However, unless there's a very good reason, iPad applications should always rotate to and launch from any orientation. So, let's fix that.

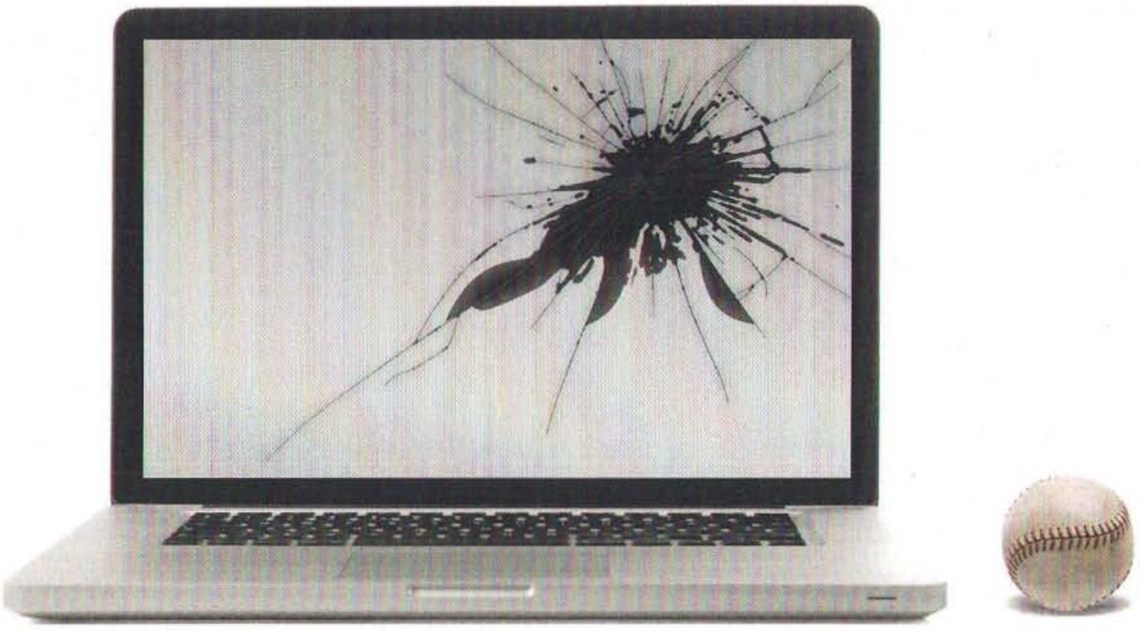
In both the `RootViewController.m` and `DetailViewController.m`, modify `shouldAutorotateToInterfaceOrientation:` as shown below.

shouldAutorotateToInterfaceOrientation:

```
- (BOOL)shouldAutorotateToInterfaceOrientation:
        (UIInterfaceOrientation)interfaceOrientation
{

    // iPad works in any orientation
    if (UI_USER_INTERFACE_IDIOM() ==
UIUserInterfaceIdiomPad) {

        return YES;
    }

    // iphone should operate in portrait mode
    return (interfaceOrientation ==
UIInterfaceOrientationPortrait);
}
```

Here, we check to see if we're using an iPad. If we are, we allow rotations to any orientation. If not, we only allow it to operate in portrait mode. Ok, I admit it. This is pretty stupid. I mean, why wouldn't you want the iPhone to support rotations as well? Still, this lets us demonstrate the Apple-recommended approach to running device-specific code. Simply call the `UI_USER_INTERFACE_IDIOM()` macro and check it against `UIUserInterfaceIdiomPad`. If they are equal, you are on an iPad running 3.2 or later. Otherwise, you're on an iPhone or iPod Touch.

In many cases, you will actually want to use a more-fine-grained approach. We already saw one technique for testing to see if features of the base SDK are available. You can also test to see if the device supports features like the camera, GPS or gyroscope. In these cases, it doesn't really matter what type of device you have—if it has the feature, it should use it. However,

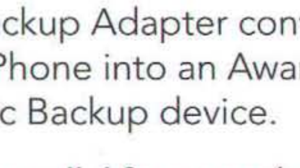


WHEN LIFE THROWS YOU A CURVE BALL.

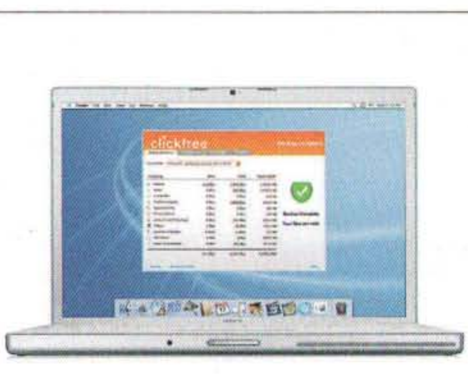
Clickfree's Transformer SE Backup Adapter converts any brand USB hard drive, iPod or iPhone into an Award-Winning, Clickfree Automatic Backup device.

Watch the demo video - www.clickfree.com/sedemo

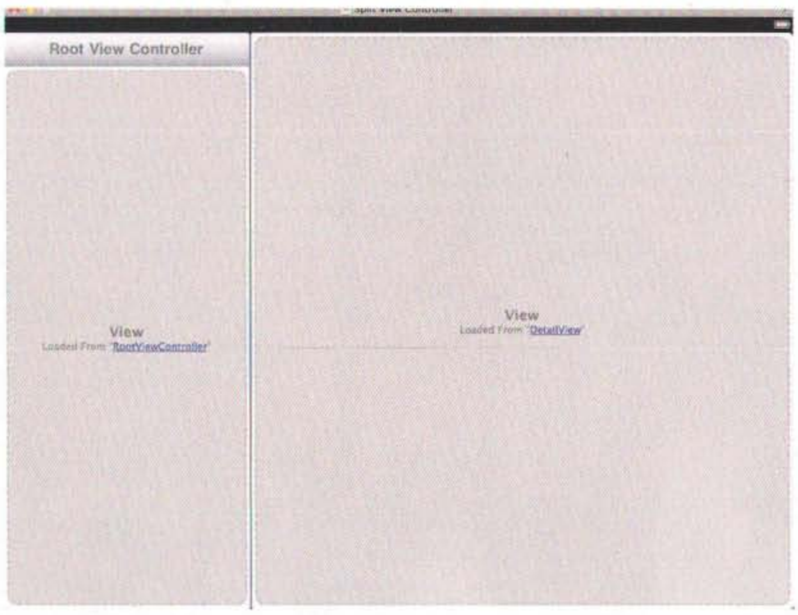
clickfree™ *Automatic Backup*



1 Plug it into your Computer with any external hard drive



2 Backup Starts Automatically - Your Files Are Safe!



Finished Split View

in this case we explicitly want different behavior between the iPad and the iPhone/iPod touch, so `UI_USER_INTERFACE_IDIOM()` is the way to go.

Note: When the iPad SDK first came out, this approach caused some trouble. It works fine on the devices, but the simulator did not support it. If you search the Internet, you will find a number of hacks to work around this problem. My advice is to just ignore them. `UI_USER_INTERFACE_IDIOM()` works just fine on the current Simulator, and will run successfully on all devices.

Build and run both the iPhone and iPad versions. Try rotating them. The iPad will rotate the UI to match, while the iPhone will not. So far so good, but we also want to launch the iPad app into any orientation. Currently, it will launch in portrait mode and then rotate to the correct orientation. Let's fix that as well.

Open `SimpleRSS-Info.plist`. The editor should display a table of key/value pairs. Select the last item on the list. You should see a + button hanging off the lower right corner of the table. Press the + button and select Supported interface orientations (iPad). You may need to widen the Key column to see the entire name.

Supported interface orientations (iPad) takes an array of values, with each value indicating a valid launch orientation for the application. These settings are specific for the iPad. You could, alternatively, set the default value: Supported interface orientations, or set the iPhone-specific values: Supported interface orientations (iPhone). But for now, we'll leave those values alone—forcing non-iPad devices to launch in portrait mode.

To edit the array of values, toggle the expansion triangle at the left of the key name. Portrait (bottom home button) was added by default. Press the + button three more times to add the other orientations: Portrait (top home button), Landscape (left home button) and Landscape (right home button). That's it. Build and run the application. Test it both in the simulator and on the device. You should be able to both launch and rotate the application to any orientation on the iPad. On the other hand, the iPhone behavior remains unchanged.

Adding A Split Screen

As you can see, porting the iPhone interface over to the iPad does not make effective use of the larger screen size. Or, to put it another way, full-screen table views are ugly. Fortunately, we can make this look like a real iPad app by adding a split view controller.

In an ideal world, we could just open `MainWindow-iPad.xib`, delete the navigation controller, drop in a split view controller, reconnect everything, and we'd be done. Of course, nothing is that simple. Apparently there's a bug in the current version of Interface Builder. It does not recognize `MainWindow-iPad.xib` as a valid iPad NIB file. So, when you open `MainWindow-iPad.xib`, the library does not include the split view controller as an option.

Hopefully, this will be fixed in future Xcode releases. For now, we can work around this by building our own NIB from scratch. Delete `MainWindow-iPad.xib` from the project, and then add a new Window XIB. Be sure to set the Product to iPad, and name it `MainWindow-iPad.xib` (overwriting existing files, if necessary).

Open this file. It should already contain a File's Owner, First Responder and Window objects. Make the following changes:

- Select the File's Owner, then select the Inspector's Identity tab. Change the File's Owner class to `UIApplication`.
- Drag an Object from the Library. Name it App Delegate, and change its type to `SimpleRSSAppDelegate`.
- Link the App Delegate object to the File's Owner's `delegate` outlet. Right click on File's Owner, and drag from the drop-down window's `delegate` circle to the App Delegate icon.
- Similarly, connect the App Delegate's `window` outlet to the Window icon.

Next, drag out a Split View Controller. Double click the controller to edit the split view interface. This should show the blank interface skeleton in landscape orientation, with a narrow view on the left and a wider view on the right.

Click on the left view, and select the Inspector's Attribute's tab. Set the NIB Name to RootViewController. Similarly, set the right view's controller to DetailView. Now, switch back to the Inspector's Identity view, and set the view controller types to `RootViewController` and `DetailViewController`, respectively. The interface should now appear as shown above:

As long as you re-used the original name for the iPad's main window NIB, you won't need to change anything in the info.plist file. However, we still need to save a reference to the split view controller somewhere in our app delegate. We cannot reuse the `navigationController` outlet, since it is the wrong type. We could change the outlet to a more-generic type—but instead, let's add a new **mainViewController** outlet. This will allow us to continue using the **navigationController** for navigation-specific behaviors on the iPhone.

Open **SimpleRSSAppDelegate.h** and modify it as shown below:

SimpleRSSAppDelegate.h

```
#import <UIKit/UIKit.h>

@interface SimpleRSSAppDelegate : NSObject
<UIApplicationDelegate> {

    UIWindow *window;

    UIViewController *mainViewController;
    UINavigationController *navigationController;
}

@property (nonatomic, retain) IBOutlet UIWindow *window;

@property (nonatomic, retain) IBOutlet
    UIViewController *mainViewController;

@property (nonatomic, retain) IBOutlet
    UINavigationController *navigationController;

@end
```

Here we simply add an instance variable and outlet for our main view controller. In **SimpleRSSAppDelegate.m**, be sure to add the code to synthesize and deallocate the new **mainViewController**. Then change **application:did-FinishLaunchingWith-Options:** as shown below.

SimpleRSSAppDelegate.m

```
-(BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions {

    // Override point for customization after app launch

    [window addSubview:[self.mainViewController view]];
    [window makeKeyAndVisible];

    return YES;
}
```

Here, we add the **mainViewController** (not **navigationController**) to the app's window. As long as we set the proper controller in both **MainWindow-iPad.xib** and **MainWindow.xib**, the same code will work on both devices.

Open **MainWindow-iPad.xib** again. Connect the App Delegate's **mainViewController** outlet to the Split View Controller icon. Leave the **navigationController** outlet blank. Similarly, in **MainWindow.xib**, connect the **mainViewController** to the Navigation Controller icon, but in this case, leave the **navigationController** outlet connected as well.

Build and run the application. It should still work properly on the iPhone. It also runs on the iPad—more or less.

We still have two bugs. First, when it's in portrait orientation, there is no way to view the list of articles. You must rotate it to landscape mode to select an article. More importantly, when you do move to landscape orientation and select an article, instead of filling the detail view, the web view squeezes into the left column and replaces the list of articles.

On the one hand, this makes sense. After all, this is exactly what happens on the iPhone, and we haven't touched that code yet. However, we never added a navigation controller to the split-view's left column, so why does this code work at all?

On closer inspection , it turns out that the left column comes with a table view controller and navigation bar pre-loaded inside a navigation controller. When we set the NIB name and type, we simply re-defined this table view controller. It's still nestled inside its navigation controller. While we won't use the navigation controller in this project, knowing it's there will greatly simplify projects that navigate through multi-level hierarchical data. But be careful. The right column is just a simple view without any navigation support at all. If you want a navigation controller there as well, you will have to add it yourself.

So, let's address these problems in reverse order. First off, **RootViewController** needs a way to communicate with the correct detail view. If you look at the source code, you will see that **RootViewController** already has a **detailViewController** instance variable. This variable is lazily initialized the first time a row is selected, and is then pushed onto the navigation controller's stack. This obviously works great for the iPhone, but in the iPad we want to manually set the **detailViewController** to the version loaded in the NIB file, and we don't want to push it onto the navigation stack.

Let's start by adding an **IBOutlet** property for the controller in **RootViewController.h**:

RootViewController.h

```
#import <UIKit/UIKit.h>

@class DetailViewController;

@interface RootViewController : UITableViewController {

    NSArray* articles;
    DetailViewController* detailViewController;

}

@property (nonatomic, retain) IBOutlet
    DetailViewController* detailViewController;

@end
```

Now, turning to the implementation file, you will see that a detailViewController property is already defined in the extension. If you haven't seen extensions before, they are a convenient way to define private methods. Since we just publicly declared this outlet, we no longer need the private declaration. Simply delete the detailViewController property

STOP SHARING!

START FAXING!

Each subscriber receives faxes directly by email as PDF file attachments.

Corporate accounts from 3 to 100+ users available

For more information and a special offer for MacTech readers, visit

www.MaxEmail.com/MacTech

maxemail®

Call: 800-964-2793

from the extension. We've already synthesized it and deallocated it, so everything else should still work properly.

Next, modify tableView:didSelectRowAtIndexPath: as shown below:

tableView:didSelectRowAtIndexPath:

```
- (void)tableView:(UITableView *)tableView
    didSelectRowAtIndexPath:(NSIndexPath *)indexPath {

    // make sure we have a detail view controller
    if (self.detailViewController == nil) {

        DetailViewController* controller =
            [[DetailViewController alloc]
               initWithNibName:@"DetailView" bundle:nil];

        self.detailViewController = controller;
        [controller release];
    }


    // Then have the controller load the URL.
    Entry* article = [self.articles
        objectAtIndex:indexPath.row];

    [self.detailViewController loadURL:article.link];

    // if we set the navigation controller, use it.
    id appDelegate =
        [[UIApplication sharedApplication] delegate];

    if ([appDelegate navigationController]) {
        [self.navigationController
            pushViewController:self.detailViewController
            animated:YES];
    }

}
```

Save this file and go back to the `MainWindow-iPad.xib`. Open the split view and right click on the left column (RootViewController). Connect the `detailViewController` outlet to the right-column (DetailView). Build and run. The detail view should now display properly (more or less—the ActivityHUD is offset a bit, but we'll fix that when we add the Article List button).

There are two main things going on here. On the iPhone, `detailViewController` is not set by the NIB. That means its initial value is `nil`, so our lazy initialization code still creates a new controller the first time a row is selected. On the iPad, we have already set the `detailViewController` in the NIB, so the initialization code is never run. As a result, we get the correct behavior on both devices.

Next, we could check to make sure we're not on an iPad before pushing the detail view onto the navigation controller—but, as we saw earlier, it's best to make your conditional code as fine-grained as possible. Here we'll use the following rule of thumb. If we've defined a navigation controller, use it. Otherwise don't. That way (hopefully) our code will be easier to extend should Apple release any additional devices.

Now, as we said before, the right column on a split view is automatically placed in a navigation controller, so we cannot just check `self.navigationController`. Instead, look at the App Delegate's `navigationController` outlet. We set that outlet in the iPhone NIB, but not in the iPad's. So, we check it instead.

Adding the button's a bit more complicated. We'll need to add a toolbar, which means creating a new NIB for the iPad. However, we can still use the same `DetailViewController` for both NIBs. Make the following changes:

Right click on Resources-iPad, and add a new View XIB. Be sure to select the iPad product, and then name the NIB `DetailView-iPad`.

Copy the activityHUD from our old detail view. Simply open `DetailView.xib` and drag and drop the activityHUD icon into the new NIB file.

Open the view and add a Toolbar across the top. Then fill the bottom of the view with a Web View (make sure it fills the entire view, on my laptop, I have to scroll down to get to the bottom).

Delete the toolbar's item button, and make sure the toolbar and web view will resize properly. You probably also want the web view to Scale Pages To Fit (Attributes tab).

Set the File's Owner's class to `DetailViewController`.

Now, we need to make a few changes to the `DetailViewController` before we link things up. Let's start with `DetailViewController.h`. Modify it as shown below:

DetailViewController.h

```
#import <UIKit/UIKit.h>

@interface DetailViewController : UIViewController
    <UIWebViewDelegate,
     UIPopoverControllerDelegate,
     UISplitViewControllerDelegate> {

    UIActivityIndicatorView* activityIndicator;
    UIView* activityHUD;
    UIWebView* webView;

    // added for ipad support
    UIPopoverController *popoverController;
    UIToolbar *toolbar;

}

@property (nonatomic, retain) IBOutlet UIWebView*
webView;
@property (nonatomic, retain) IBOutlet
UIActivityIndicatorView* activityIndicator;
@property (nonatomic, retain) IBOutlet UIView*
activityHUD;
@property (nonatomic, retain) IBOutlet UIToolbar
*toolbar;

- (void)loadURL:(NSURL*)url;
```

and you thought rabbits duplicated fast

866.4.RABBIT

- CD/DVD Manufacturing
- Content Creation
- Fulfillment / Distribution
- Marketing

www.amsrabbit.com

```
@end
```

We start by adding the `UIPopoverController-Delegate` and `UISplitViewControllerDelegate` prototypes. Next, we added instance variables and outlets to support the toolbar and popover. We also add an explicit `webView` instance variable.

In our previous version, our view was itself the `UIWebView` object. We had a `webView` property that simply cast and returned the main view. In this version, `UIWebView` is a sub-view. As a result, we need to add a read/writeable outlet to set and access it.

Note: Since the popover is never used outside this class, we will define its property in a class extension (just as we saw with the `DetailViewController`). Also, be sure to synthesize and deallocate `webView`, `toolbar` and `popoverController`. You will also need to delete the existing `webView` method—we want to make sure we use the new, synthesized methods instead. I won't show these steps here, but if you have any problems, check out the final source code.

Next, modify the `loadURL:` method as shown below:

loadURL:

```
- (void)loadURL:(NSURL*)url {

    NSURLRequest* request = [NSURLRequest
requestWithURL:url];
    [self.webView loadRequest:request];

    // dismiss the popover (if any exist) when a view
is loaded
    if (popoverController != nil) {
        [popoverController dismissPopoverAnimated:YES];
    }

}
```

Here, we simply add code to see if we are displaying the popover view. If we are, we dismiss it. This code works on both the iPhone and iPad. On the iPhone, `popoverController` will always be set to `nil`, so the code never executes. On the iPad, if we're using the app in portrait mode, this will clear away the popover once an article is selected.

Finally, we implement two `UISplitViewControllerDelegate` methods to enable and disable our popover list.

DetailViewController.m

```
#pragma mark -
#pragma mark SplitView and Popover Delegate Methods

- (void)splitViewController: (UISplitViewController*)svc
     willHideViewController:(UIViewController
*)aViewController

withBarButtonItem:(UIBarButtonItem*)barButtonItem
       forPopoverController: (UIPopoverController*)pc {

    barButtonItem.title = @"Article List";

    NSMutableArray *items = [[toolbar items]
mutableCopy];

    [items insertObject:barButtonItem atIndex:0];
    [self.toolbar setItems:items animated:YES];
    [items release];

    self.popoverController = pc;
}


// Called when the view is shown again in the split
view, invalidating the button and popover controller.
- (void)splitViewController: (UISplitViewController*)svc
     willShowViewController:(UIViewController
*)aViewController
  invalidatingBarButtonItem:(UIBarButtonItem
*)barButtonItem {

    NSMutableArray *items = [[toolbar items]
mutableCopy];
    [items removeObjectAtIndex:0];

    [self.toolbar setItems:items animated:YES];
    [items release];

    self.popoverController = nil;
}
```

I find the names a bit confusing. The first method will be called when the split view is rotated into portrait mode (thus "hiding" the split view). It provides us with a toolbar button and a popover that will contain our list of articles. We simply insert the button into the toolbar, and save a reference to the popover controller.

The second method will be called when the view is rotated back (thus, "showing" the split view). Here, we get rid of the toolbar button and the popover controller.

Now we need to link everything together. Open `MainWindow-iPad.xib` and make the following changes:

Change the right column so the NIB Name is `DetailView-iPad`. You don't need to change the type, since it's still using the `DetailViewController`.

Connect the Split View Controller's `delegate` outlet to the DetailViewController (right column).

Now, open `DetailView-iPad.xib`.

Link the File's Owner's `activityHUD` and `view` outlets to the corresponding icons.

Open the ActivityHUD and link the `activityIndicator` to the white indicator in the middle of the view.

Open the View, and link the `toolbar` and `webView` outlets to their corresponding objects.

We also need to fix the iPhone version. Open `DetailView.xib`, and link the File's Owner's `webView` outlet to the Web View icon.

Build and run the application. The iPad version should now work in both portrait and landscape mode. However, if you load the iPhone version, you'll notice that it displays a blank page the first time you select an article. Going back and selecting a new article works properly.

SendStuffNow™

Secure Online Delivery for All Your Files

Send the Large Files that Email Can't Handle

Go to **offers.sendstuffnow.com/mactech** and Sign Up for Your Free Account Today!

SMITHMICRO SOFTWARE

Visit us online at **sendstuffnow.com** or call **1-877-781-8284**

We let a rather subtle bug slip in here. Look again at RowViewController's didSelectRowAtIndexPath: method. It's important to realize that when we create a new DetailViewController, initWithNibName: does not actually return a completely initialized controller. Instead, much of the initialization is delayed until the DetailViewController.view property is accessed.

Normally this is done automatically (for example, when you push the view controller onto the navigation stack). However, we call loadURL: right after initializing the new controller. This, in turn, calls self.webView and loads the given URL. In our previous version, self.webView was simply a call to self.view—so everything was set up properly before loadRequest: got called. Now, however, self.webView will be set by the NIB. We haven't called self.view yet, so it will still be set to nil, and loadRequest: fails silently.

Fortunately, there is a simple fix. Make sure the view is pushed onto the navigation stack before we load the url, as shown below:

RootViewController's tableView:didSelectRowAtIndexPath:

```
- (void)tableView:(UITableView *)tableView
    didSelectRowAtIndexPath:(NSIndexPath *)indexPath {

    // make sure we have a detail view controller
    if (self.detailViewController == nil) {

        DetailViewController* controller =
[[DetailViewController alloc]
            initWithNibName:@"DetailView" bundle:nil];

        self.detailViewController = controller;
        [controller release];
    }

    // if we set the navigation controller, use it.
    id appDelegate = [[UIApplication sharedApplication]
delegate];

    if ([appDelegate navigationController]) {
        [self.navigationController

pushViewController:self.detailViewController
animated:YES];
    }

    // Then have the controller load the URL.
    Entry* article = [self.articles
objectAtIndex:indexPath.row];
    [self.detailViewController loadURL:article.link];

}
```

However, this brings up a very important point. When you're working on a universal application, you must continually test all supported devices—not just the device you're currently working with. Sometimes minor changes can have unexpected side effects.

Changing the Feed, Fixing Splash Screens and Other Small Details

The main functionality is now in place, but there are a few details we should clean up. First, we want the iPad version to load the iPad support feed, not the iPhone feed. I'll leave this as a homework assignment; however, you should be able to use the UI_USER_INTERFACE_IDIOM() macro, just like we did before.

Next, we want to load an appropriate splash screen for each device. We already have a portrait splash screen for the iPhone. We need both a portrait and landscape splash screen for the iPad. The portrait splash screen should be a 768 x 1004 pixel PNG file, while the landscape version should be 1024 x 748. Save these as SimpleRSS_IpadSplash-Portrait.png and SimpleRSS_IpadSplash-Landscape.png, and add them to the project. If you wish, you can also add an iPhone 4 version (640 x 960 pixels) and save it as SimpleRSS_Splash@2x.png.

Now open SimpleRSS-Info.plist. The default launch image is already set for SimpleRSS-Splash. Add a new Launch Image (iPad) key, and set the value to SimpleRSS_IpadSplash. The application will now automatically use the correct image based on the device's orientation and the file names.

There are a couple of interesting points here. First, instead of setting device specific launch images in the info.plist, in theory we could just add ~iPhone and ~iPad to the file names. However, I had trouble getting the ~iPhone and ~iPad tags to play nice with the –Landscape and –Portrait tags. Defining a device specific base file name, then just using the orientation tags worked a lot better for me.

Also, Xcode tries to avoid compiling and sending new data to the device or simulator unless it really has to. Sometimes, simply changing a resource or the info.plist isn't enough to actually change the app. When I was testing the new splash screens, my changes often didn't make it to the device. When this happens, it's usually best to delete the app from the device (or simulator), then clean all targets and rebuild. This guarantees that a new version will get built and loaded.

Also, remember that closing an app on iOS 4.0 does not actually exit the application. This can cause some confusion when you launch the app and it fast switches in instead of showing the splash screen. Just force quit the app, and try again.

Finally, we need to add specific application and search icons for each device. For the iPhone, this is a 57 x 57 main icon and a 29 x 29 search icon (114 x 114 and 58 x 58 for iPhone 4). For the iPad these are 72 x 72 and 50 x 50.

Icons are handled somewhat differently in the info.plist. If you open SimpleRSS-Info.plist, you will see an Icon Files key with an array of icon names (**Note:** this is different from the Icon File key). It should already have the SimpleRSS_Icon and SimpleRSS_SearchIcon entries. Create PNG files for your iPad's icons. Add them to the project, and then add their names (without the .png extension) to the array. Here, the names don't matter; iOS will automatically figure out which icon to use based on their size.

Conclusion

That's it. We have a Universal application that runs on both the iPhone and iPad. Despite being a relatively straightforward conversion, there are still a lot of steps involved in getting everything working properly. Real applications will take even more effort. So, plan ahead.

Don't forget your old-school software engineering. In this project, we tended to use conditionals to determine when device specific code should be run. However, this can quickly become hard to read and maintain. In most cases, you are probably better off creating a single super class that contains all the common functionality, and then create separate sub-classes for each specific device. In many cases, you can set up your MainWindow NIBs to load the proper sub-classes, and you can completely do away with the runtime conditionals.

Also, we covered a lot of territory in a relatively short time—especially there at the end. I will include copies of both the original and the final source code at ftp.mactech.com/src/mactech/volume26_2010/26.08.sit/. If you have any questions or problems, just open the final source code and check out the results (remember: the FileMerge utility is your friend—use it to compare original vs. final source files).

Good luck, and happy coding. Now go make something cool.

MT

About The Author

Rich Warren lives in Honolulu, Hawaii with his wife, Mika, daughter, Haruko, and his son, Kai. He is an associate scientist with 21st Century Systems, Inc., a freelance writer and a part time graduate student. When not playing on the beach with his kids, he is probably writing, coding or doing research on his MacBook Pro. You can reach Rich at rikiwarren@mac.com, check out his blog at http//freelancemadscience.blogspot.com or follow him at http//twitter.com/rikiwarren

Introducing a better hosting solution.

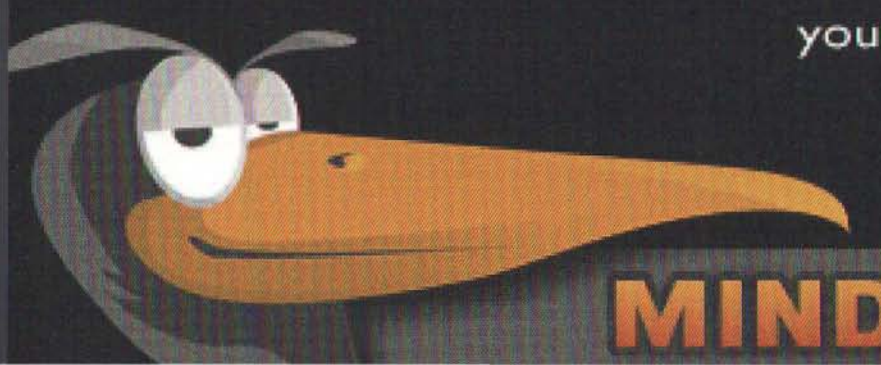
✓ Running CloudLinux
✓ Limited Accounts Per Server
✓ Full cPanel With Each Account
✓ Daily Backups
✓ Instant Setup!

Discounts on our UBB Forum

- In use since 1997
- Easy to set up
- Easy to administer
- Built in Photo Gallery

As seen on the MacTech Forums at applecentral.com

Visit www.mindraven.com and let us show you how we can give your website a better home.



MINDRAVEN.com

Managing Firefox, Revisited

Managing Firefox Settings for your organization

By Greg Neagle, MacEnterprise.org

MacEnterprise.org
Mac OS X enterprise deployment project

Introduction

Back in the December 2008 issue of *MacTech*, we looked at managing Firefox in an enterprise environment. Much has changed in the past two years, and it's time to take a look again at this topic.

Mac users have a choice of several excellent browsers. Safari, included with Mac OS X, is a great browser. Google has made its fast WebKit-based browser, Chrome, available on the Mac, and Firefox is still very popular. All three browsers continue to compete against each other on speed, support for HTML5, and other features. This competition causes all three browsers to continue to improve. The winner is the Mac user.

If your organization uses Firefox, you may need to manage its configuration. You might want to set a default home page that is unique to your organization. You may need to configure proxy settings so that Firefox can actually reach the Internet from inside your network. If you are managing software installs and updates, you might want to turn off these features inside Firefox.

For many applications, the solution for managing these sorts of things is to use Apple's Managed Preferences, or MCX. But Firefox does not store its important configuration files in an Apple-style plist. So you need to use other techniques to manage Firefox preferences.

The Old Way

In the December 2008 MacEnterprise column, I presented one technique for managing Firefox preferences. This technique involved editing some files inside the Firefox application bundle, and adding another file inside the bundle. That column is available online here:

http://www.mactech.com/articles/mactech/Vol.24/24.12/2412MacEnterprise/

While the technique described in that past column still works, there are some issues with this approach. Since files are added and/or modified inside the application bundle, every time Firefox is updated, you need to either repackage Firefox with your changes, or find a way to re-apply your modifications on top of the new Firefox application after it is installed. Though this might not be terribly difficult or time-consuming, the best systems administrators are always looking for ways to eliminate boring, repetitive work like this. I wanted to find a way I could install our modifications once, and in a way that later updates to Firefox would not need repackaging or reinstall of our customizations.

A New Way

Mozilla has a tool called the "Client Customization Kit" or "CCK" which was designed to allow organizations to customize Firefox for their use. An independent developer, Michael Kaply, has created a CCK Wizard, which provides a wizard-style interface to help you create a Firefox extension to include the customizations in Firefox.

I had looked at the Client Customization Kit in the past, but for a variety of reasons had thought it would not be able to do what I needed. One big problem I saw was that the CCK creates a Firefox extension. These are usually installed in the user's profile in his or her home directory, so that's not a good fit for something you want to manage enterprise-wide. With Firefox 2.x and earlier, there is a way to install extensions "globally", but they end up inside the application bundle on OS X, and that's the same problem I was trying to solve – every time I updated Firefox, I'd have to repackage Firefox or reinstall the extension into the updated application's bundle.

Firefox 3 and Global Extensions

With the release of Firefox 3.x, a new location for global extensions was added. The location varies depending on the OS — Mac OS X, Windows, or Linux — but the important thing is that this new location is *outside* of the application bundle on OS X. This new global extension location is:

```
/Library/Application Support/Mozilla/Extensions/{ec8030f7-c20a-
464f-9b0e-13a3a9e97384}/
```

You can learn more about installing extensions for Firefox 3 and related applications here:

https://developer.mozilla.org/en/Installing_extensions

Since the new location for global extensions is outside the Firefox application bundle, it survives Firefox updates with no additional work needed. This makes it possible to:

Use the Firefox CCK to create a Firefox extension that customizes Firefox settings for your organization, and then:

Build an installer that puts this extension in the right place to be used by all Firefox users on a given machine. You can then install this once, and future updates of Firefox will not require this to be reinstalled.

Hopefully, Firefox settings for your organization change less frequently than Firefox itself does, meaning that this will be less work overall. Some other advantages (or least changes) that result from this approach:

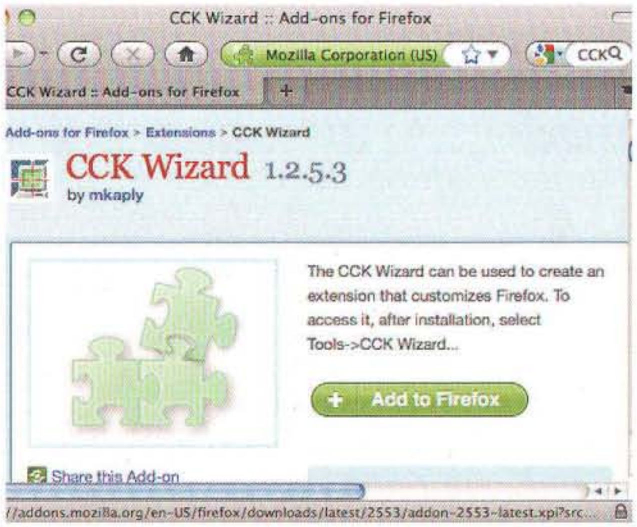
Users with administrative rights can update Firefox themselves and still have your organization's settings applied. Previously, if they had replaced your customized Firefox with a version downloaded from mozilla.com, the custom configuration would be erased.

Some additional settings become much easier to manage; for example, the "feature" that takes you to a "What's New" page with each new release of Firefox is easy to turn off now. That was possible before with manual configuration, but now it's a check box in the CCK Wizard.

Users who really dislike your custom settings or want/need to turn them off for some reason can now do so. In Firefox, select Add-ons from the Tools menu. In the Add-ons window, select the Extensions tab, then your CCK extension, and click Disable. If you don't want your users to be able to disable your extension, that can be managed as well.

CCK Wizard Walkthrough

Let's walk through using the CCK Wizard to create an extension that customizes Firefox for your organization. Start by launching Firefox and go to the CCK Wizard page at https://addons.mozilla.org/en-US/firefox/addon/2553/. Install the CCK Wizard extension by clicking the Add to Firefox button as shown in Figure 1.

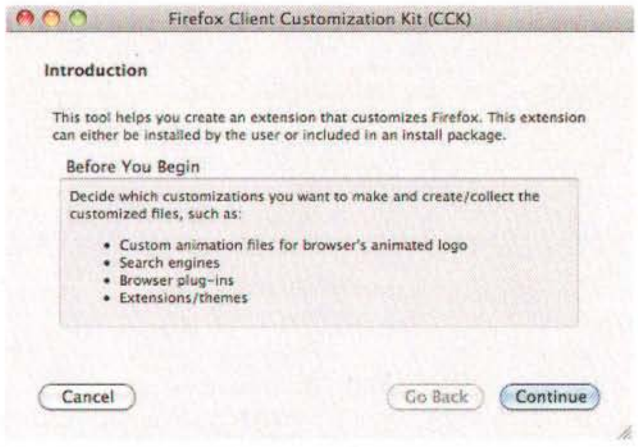


The CCK Wizard can be used to create an extension that customizes Firefox. To access it, after installation, select Tools->CCK Wizard...

/addons.mozilla.org/en-US/firefox/downloads/latest/2553/addon-2553-latest.xpi?src...

Figure 1 – Installing the CCK Wizard

You'll see a warning telling you to "install add-ons only from authors whom you trust." Click Install Now. After a few moments you'll be prompted to restart Firefox to complete the installation. Once Firefox relaunches, you can start the CCK Wizard from the Tools menu, where you'll see a new CCK Wizard item. You'll see the CCK Wizard Introduction screen in Figure 2.



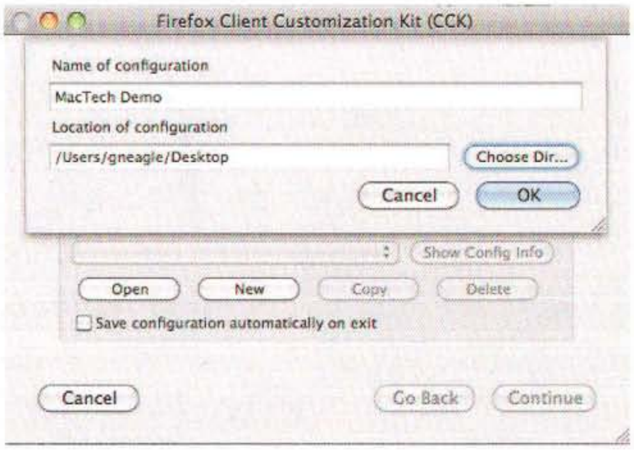
This tool helps you create an extension that customizes Firefox. This extension can either be installed by the user or included in an install package.

Decide which customizations you want to make and create/collect the customized files, such as:

- Custom animation files for browser's animated logo
- Search engines
- Browser plug-ins
- Extensions/themes

Figure 2 – CCK Introduction

Click Continue. In the next screen, create a new configuration. Figure 3 shows an example.



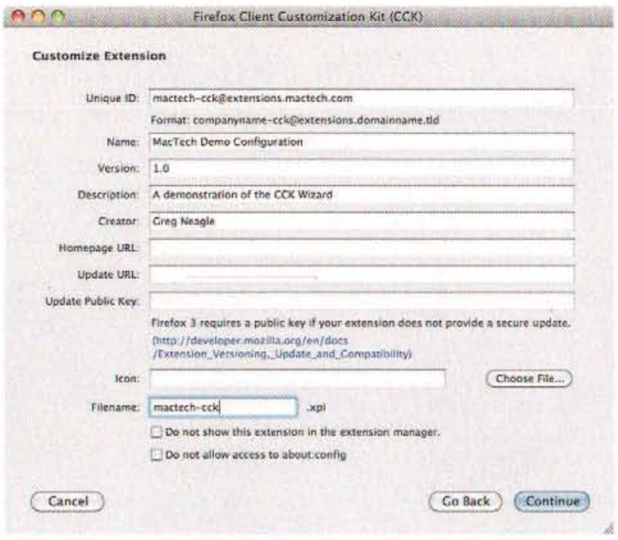
Name of configuration: MacTech Demo

Location of configuration: /Users/gneagle/Desktop

☐ Save configuration automatically on exit

Figure 3 – Creating a new configuration

Click Continue. Next, you'll need to enter some identifying information for the extension you are creating. Figure 4 illustrates this.



Customize Extension

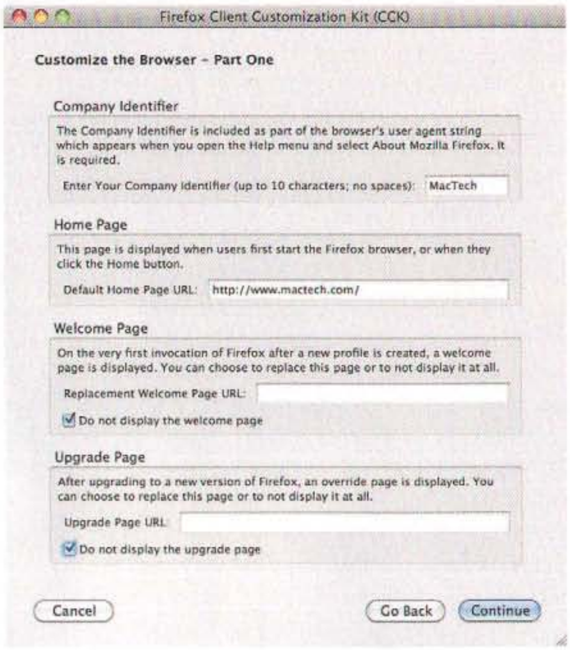
Unique ID: mactech-cck@extensions.mactech.com
Format: companyname-cck@extensions.domainname.tld
Name: MacTech Demo Configuration
Version: 1.0
Description: A demonstration of the CCK Wizard
Creator: Greg Neagle
Homepage URL:
Update URL:
Update Public Key:
Firefox 3 requires a public key if your extension does not provide a secure update.
(http://developer.mozilla.org/en/docs/Extension_Versioning,_Update_and_Compatibility)
Icon:
Filename: mactech-cck .xpi
☐ Do not show this extension in the extension manager.
☐ Do not allow access to about:config

Figure 4 – Basic extension information

Many of the fields here are optional. In fact, only the unique ID, name, and version are required. Since you are not going to distribute this publicly, you don't need a public key or either URL. Earlier I mentioned that a user by default could disable your extension – if you want to prevent that, check the Do not show this extension… box. If the extension doesn't appear in the Extension Manager, the user can't disable it. Once you are finished entering information on this page, click Continue once again.

Let the Customizing Begin

Finally we can start customizing Firefox settings. The next few pages walk you through some of the most common settings to customize. Figure 5 shows the first of several pages.



Customize the Browser – Part One

Company Identifier
The Company Identifier is included as part of the browser's user agent string which appears when you open the Help menu and select About Mozilla Firefox. It is required.
Enter Your Company Identifier (up to 10 characters; no spaces): MacTech

Home Page
This page is displayed when users first start the Firefox browser, or when they click the Home button.
Default Home Page URL: http://www.mactech.com/

Welcome Page
On the very first invocation of Firefox after a new profile is created, a welcome page is displayed. You can choose to replace this page or to not display it at all.
Replacement Welcome Page URL:
✓ Do not display the welcome page

Upgrade Page
After upgrading to a new version of Firefox, an override page is displayed. You can choose to replace this page or to not display it at all.
Upgrade Page URL:
✓ Do not display the upgrade page

Figure 5 – Browser customization

HoudahSpot

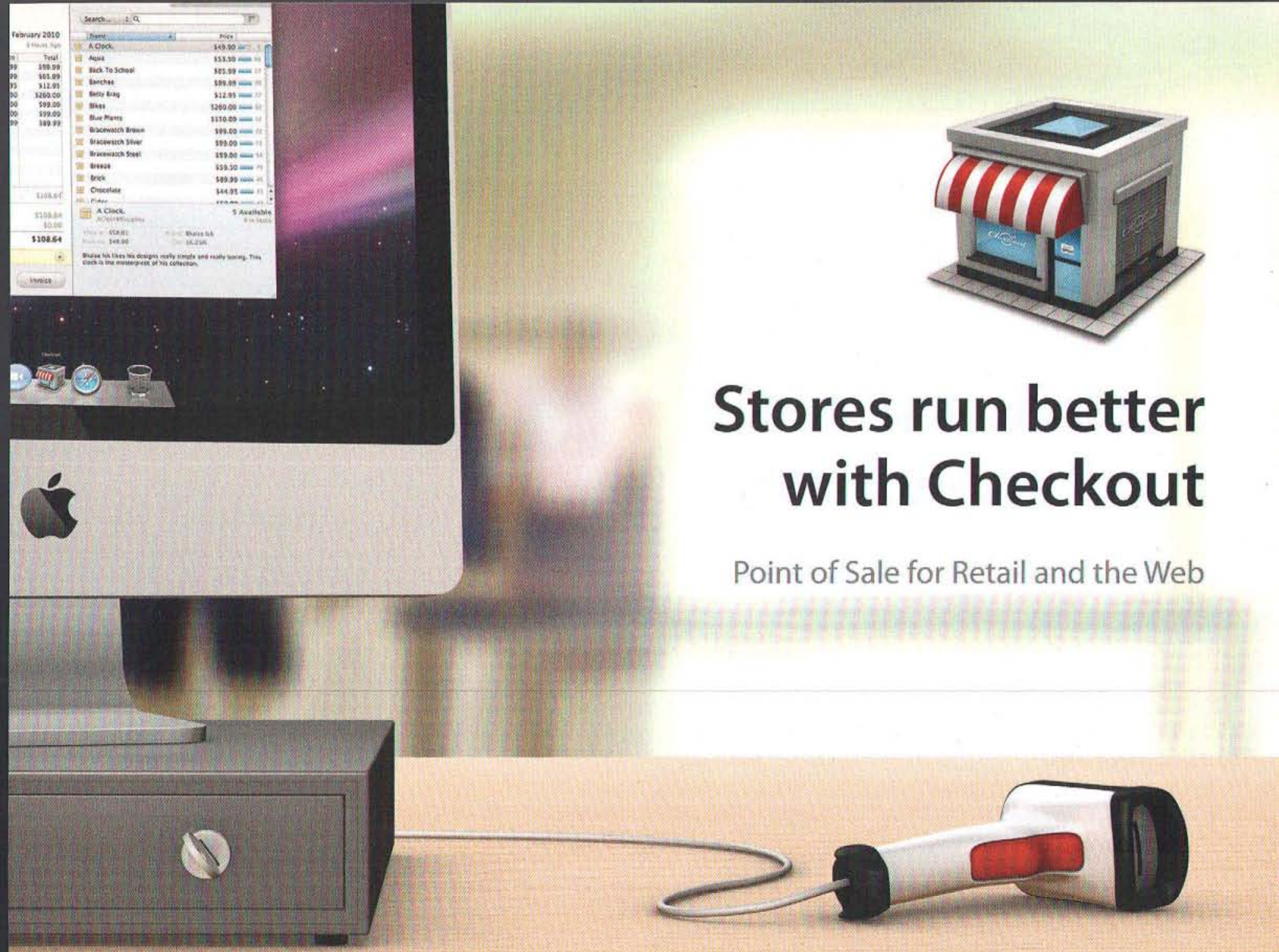
Spotlight front-end you are missing

File tagging solution. Tag search tool

"HoudahSpot is EXACTLY what Spotlight should have been in Mac OS X."
Chris Pirillo

"...give HoudahSpot a test run—you may find it to be the perfect companion for Spotlight."

Macworld



Name	Price
A Clock.	$49.90
Aqua	$53.50
Back To School	$85.99
Banches	$89.99
Betty Brag	$12.95
Bikes	$280.00
Blue Plants	$130.00
Bracewatch Brown	$99.00
Bracewatch Silver	$99.00
Bracewatch Steel	$59.00
Breeze	$59.50
Brick	$89.99
Chocolate	$44.95

February 2010
Total
$108.64
Invoice
A Clock.
5 Available
Blaise Ish likes his designs really simple and really boring. This clock is the masterpiece of his collection.

Stores run better with Checkout

Point of Sale for Retail and the Web

We all know everything just works better on a Mac®. Now point of sale does too. Do your favorite retailer a favor, tell them to try Checkout for free today.

Starting from **$399**, Complete **hardware bundles** available*

Visit **www.checkoutapp.com** or call **877 788 1202** toll free.

A brief description of the next few screens:

Customize the Browser – Parts Two through Four

These screens allow you to customize title bar text, animated logos, add a custom help menu item, and specify allowed and denied domains for popups, xpi installs, and cookies.

Customize Browser Plug-ins

This allows you to add default plug-ins. If you are managing software installations on your machines, this could just be handled by your regular software install mechanism.

Customize Search Engines

Use this page to add additional search options, and to choose the default search engine.

Customize Extensions/Themes

Add extensions or themes to be installed with the CCK.

Customize the Bookmarks Toolbar – Parts One and Two and **Customize Bookmarks – Parts One and Two**

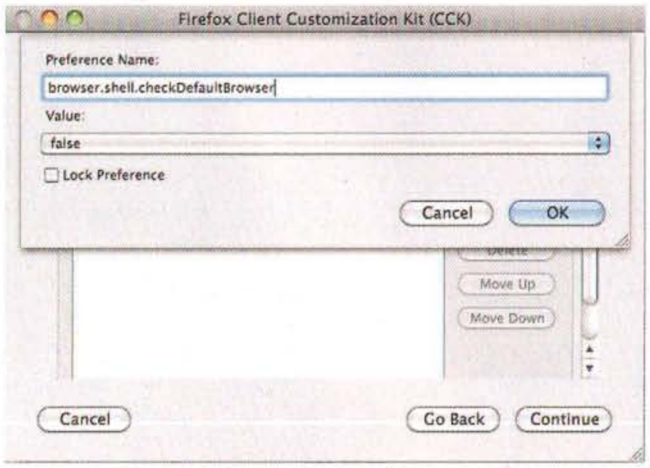
These four pages can be used to add bookmarks and bookmark folders to Firefox.

Customize Preferences

The next page needs a special mention. The CCK Wizard does not have a GUI item for every possible Firefox preference – there are literally hundreds of available preferences. For preferences that are not listed anywhere else, you can use the Customize Preferences page to add them. These preferences are in the same format as you see when you enter "about:config" into Firefox's address bar. You can learn more about "about:config" entries in this mozillaZine article:

http://kb.mozillazine.org/About:config_entries

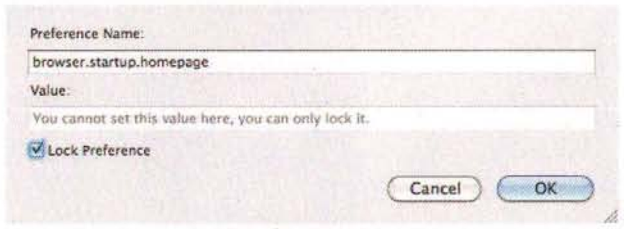
Figure 6 shows adding a preference to prevent Firefox from checking to see if it is the default browser.



Preference Name: browser.shell.checkDefaultBrowser
Value: false
☐ Lock Preference

Figure 6 – Disabling the default browser check

Note also the Lock Preference checkbox. Use this if it is important that the user not be able to change a preference. You can use this also to lock preferences that are set on other pages in the wizard, by adding them here as well and checking Lock Preference. Figure 7 demonstrates locking the home page, which was set back on the Customize the Browser – Part One page.



Preference Name: browser.startup.homepage
Value: You cannot set this value here, you can only lock it.
☑ Lock Preference

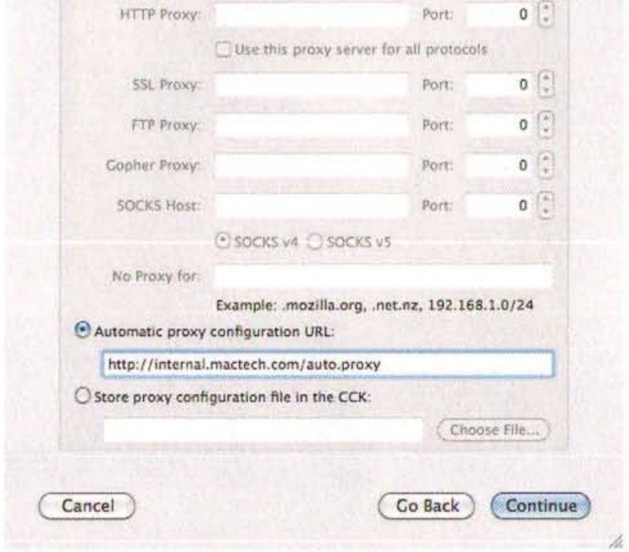
Figure 7 – Locking the home page

Since we're managing Firefox on the Mac, we can skip the next page, which allows you to add entries to the Windows registry. Following that is a page that allows you to add PEM-formatted certificates to Firefox. This might be useful to allow secure access to internal websites.

Customize Proxy Configurations

The next-to-last page of the CCK Wizard allows you to specify the proxy configuration. Note that since version 3.5, Firefox can now use the same system proxy settings that Safari uses. Figure 8 shows the options. If you need to lock any of these down, you'll need to return to the Customize Preferences page and enter the preferences there. Here are a few of the proxy preference names:

network.proxy.autoconfig_url
network.proxy.http
network.proxy.http_port
network.proxy.no_proxies_on
network.proxy.type



Customize Proxy Configurations

Configure Proxies to Access the Internet
○ No proxy
○ Auto-detect proxy settings for this network
○ Use system proxy settings
○ Manual proxy configuration:
HTTP Proxy: Port: 0
☐ Use this proxy server for all protocols
SSL Proxy: Port: 0
FTP Proxy: Port: 0
Gopher Proxy: Port: 0
SOCKS Host: Port: 0
● SOCKS v4 ○ SOCKS v5
No Proxy for:
Example: .mozilla.org, .net.nz, 192.168.1.0/24
● Automatic proxy configuration URL: http://internal.mactech.com/auto.proxy
○ Store proxy configuration file in the CCK:

Figure 8 – Proxy configuration

From print
to web in a Flash.
Work your magic
with QuarkXPress® 8

Quark

So you're a master of print design? Time to step it up a notch. Use your existing QuarkXPress skills to design for the Web and bring your creations to full interactive glory – without having to learn Flash or coding. The intuitive design interface of QuarkXPress 8 opens a world of new possibilities. Increase your productivity and offer your clients more (both print and Web), right out of the box.

But, there's more to this box of tricks than meets the eye. Buy or upgrade to QuarkXPress 8 today to access $750+ of Exclusive Flash Resources for **FREE** to make your designs shine:

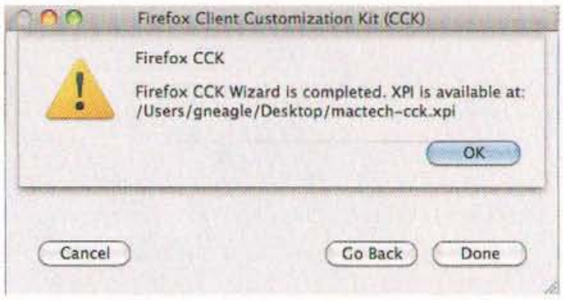
- Hundreds of exclusive, fully editable Flash assets
- Web templates, animations and video players
- Flash tutorials, eSeminars and educational resources

Upgrade NOW
for just $299 (before tax)
or BUY for $799 (before tax)

Unleash the magic
of Flash in QuarkXPress 8
visit www.quark.com/magic

Purchase
QuarkXPress® 8
today and access
$750+ worth
of Exclusive Flash
Resources for **FREE**

Finally, you reach the Conclusion page. Click Done to create the CCK package. This package is saved in the location you chose way back at the beginning, but in case you forgot, the wizard reminds you as in Figure 9.

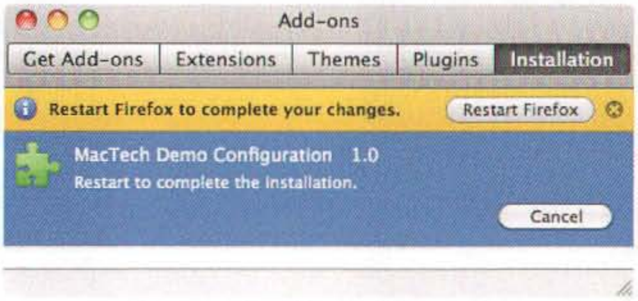


Firefox CCK Wizard is completed. XPI is available at:
/Users/gneagle/Desktop/mactech-cck.xpi

Figure 9 – CCK Wizard completed

Installing and Testing the CCK Extension

Now it's time to test the CCK extension. You can find it at the path shown at the conclusion of the CCK Wizard. Double-click it. After a warning, you should see something like Figure 10. Click Restart Firefox to finish the installation.



Restart Firefox to complete your changes.

MacTech Demo Configuration 1.0
Restart to complete the installation.

Figure 10 – Installing the CCK extension

Once it is installed, test to see if Firefox is acting as you'd expect. If you set the home page, does Firefox display it? Is it using the proxy settings you set? Are the correct preference locked? If you need to, you can open the CCK Wizard and edit the CCK extension until you are happy with it. If you make changes, you'll need to reinstall the modified extension.

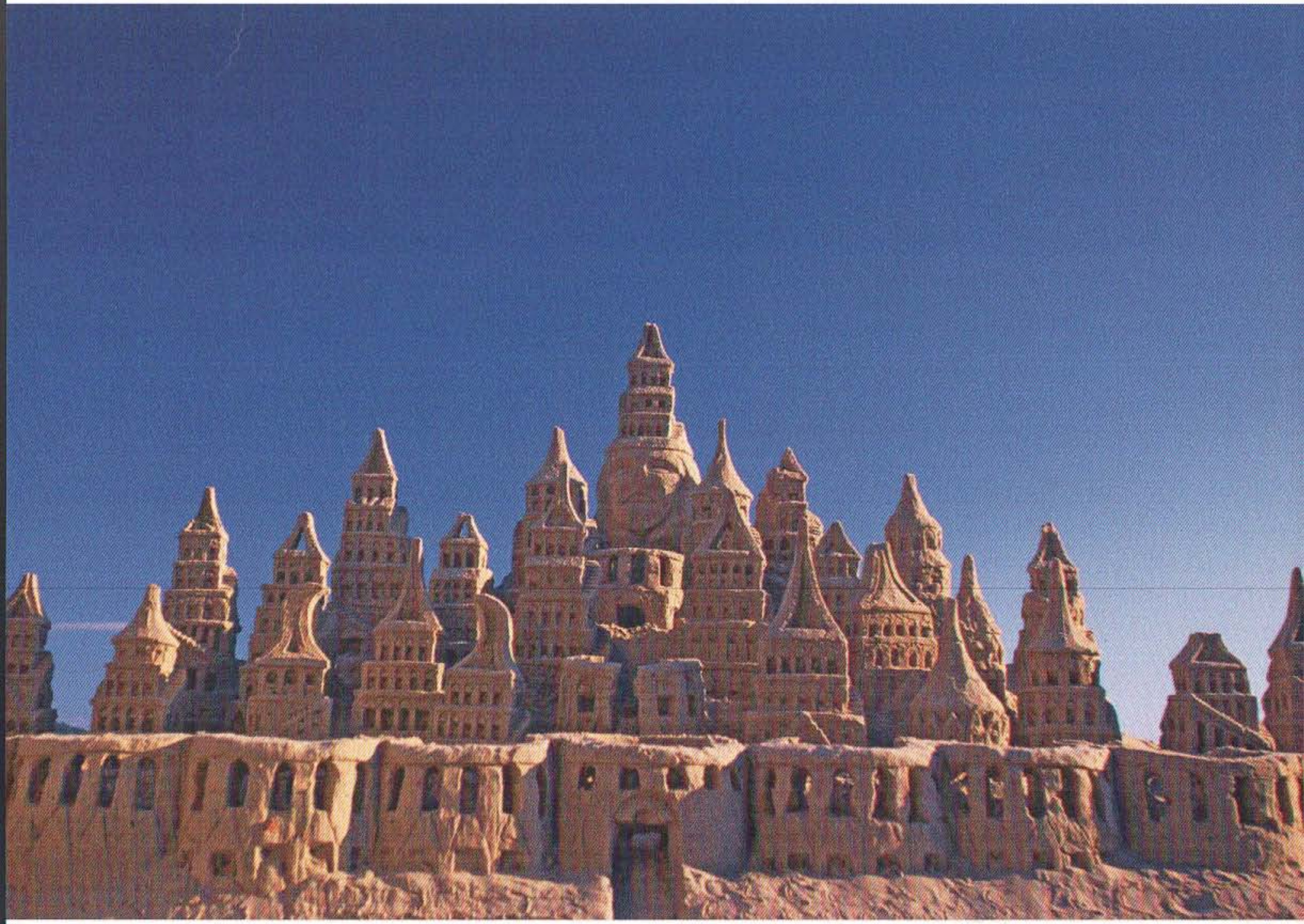
By using a double-click to install the extension, you installed it in your Firefox profile in your home directory. For deployment, you'll want to install it in the global extension location. To prepare for this, simply copy it from your profile to the global extension location.

First, locate your profile in this directory:

```
~/Library/Application
Support/Firefox/Profiles/
```

(The tilde is shorthand for your home directory.)

imagine. amaze. inspire.



you think it. you create it.

REAL Studio is the powerful, easy-to-use tool for creating your own software. At REAL Software, we call it a problem solver. You've probably said, "Wouldn't it be great if there was an application that ..." REAL Studio fills that blank.

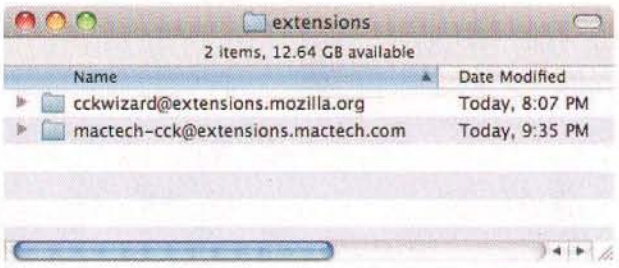
REAL Studio compiles native applications for Mac OS X, Windows and Linux from one set of source code. Each version of your software looks and works just as it should in each environment. You can even create a Universal Binary with one mouse-click.

www.realsoftware.com

Download your free 30-day trial edition today! Or buy now - REAL Studio comes with a 90-day money back guarantee.

REAL Studio

If there's more than one profile, the one you want is almost certainly the one that was modified most recently. Inside the `profile` folder is another folder named `extensions`, and the contents might look something like Figure 11.



extensions

2 items, 12.64 GB available

Name	Date Modified
cckwizard@extensions.mozilla.org	Today, 8:07 PM
mactech-cck@extensions.mactech.com	Today, 9:35 PM

Figure 11 – CCK extension in our profile directory

To make the CCK extension available to all users of this computer, quit Firefox and copy it to:

```
/Library/Application
Support/Mozilla/Extensions/{ec8030f7-c20a-464f-9b0e-
13a3a9e97384}/
```

After copying it to the global extensions directory, delete it from the Firefox profile folder in your home directory. Relaunch Firefox and test to make sure the CCK extension is working.

Final Steps

You've used the CCK Wizard to create a Firefox extension that configures Firefox for your organization. You've copied it to the global extension directory for Firefox 3.x. Now all that remains is to distribute it to all your managed machines. You can package up the contents of the CCK extension directory and use your software delivery mechanism to install it on all of your machines. After it is installed, it will take effect the next time Firefox is launched.

If you don't have a software distribution mechanism in place, you may still be able to make use of this method. In theory, you could place the XPI file created by the CCK Wizard on an internal web server or file server and direct your users to install it themselves to configure Firefox. A CCK extension done this way would affect only the user who had installed the extension, but this method provides a way for users to self-configure Firefox for use at your organization.

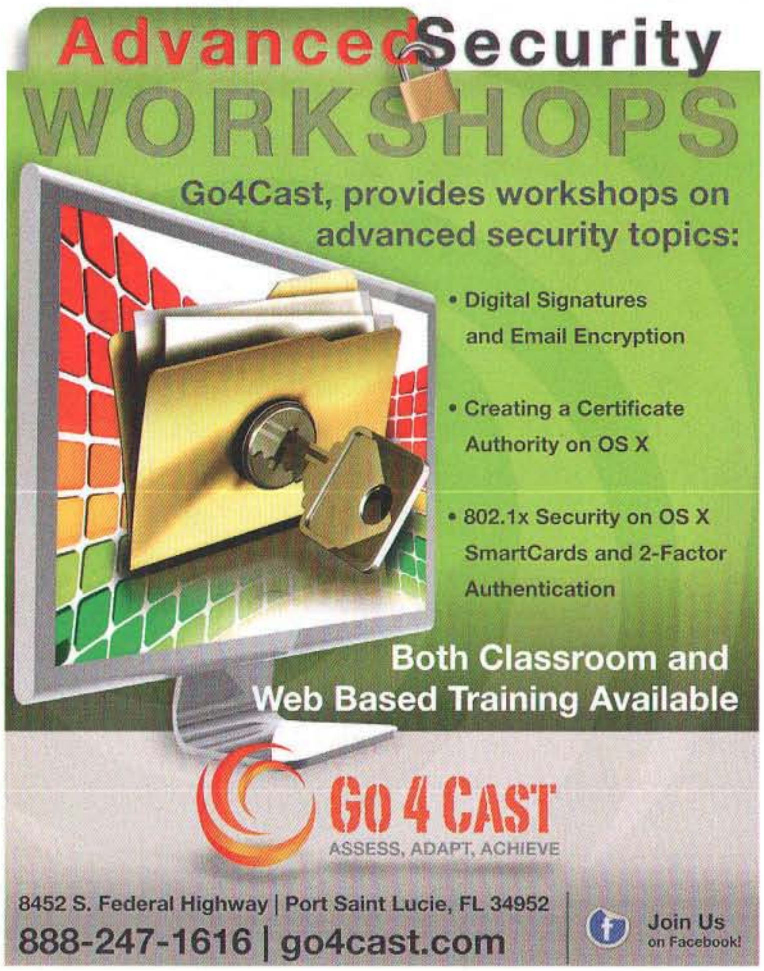
Conclusion

We've accomplished our goal, which was to implement a method to manage Firefox configuration settings that would not be affected by Firefox updates. This not only allows us to distribute newer versions of Firefox more rapidly, but also allows users with administrative rights to update Firefox themselves without losing our custom settings. The CCK Wizard makes it much easier to manage Firefox preferences than our old method of editing text files inside the Firefox application bundle. Finally, the CCK extension you create using the CCK Wizard can also be given to users directly for manual installation, making it suitable for a self-service organization.

MT

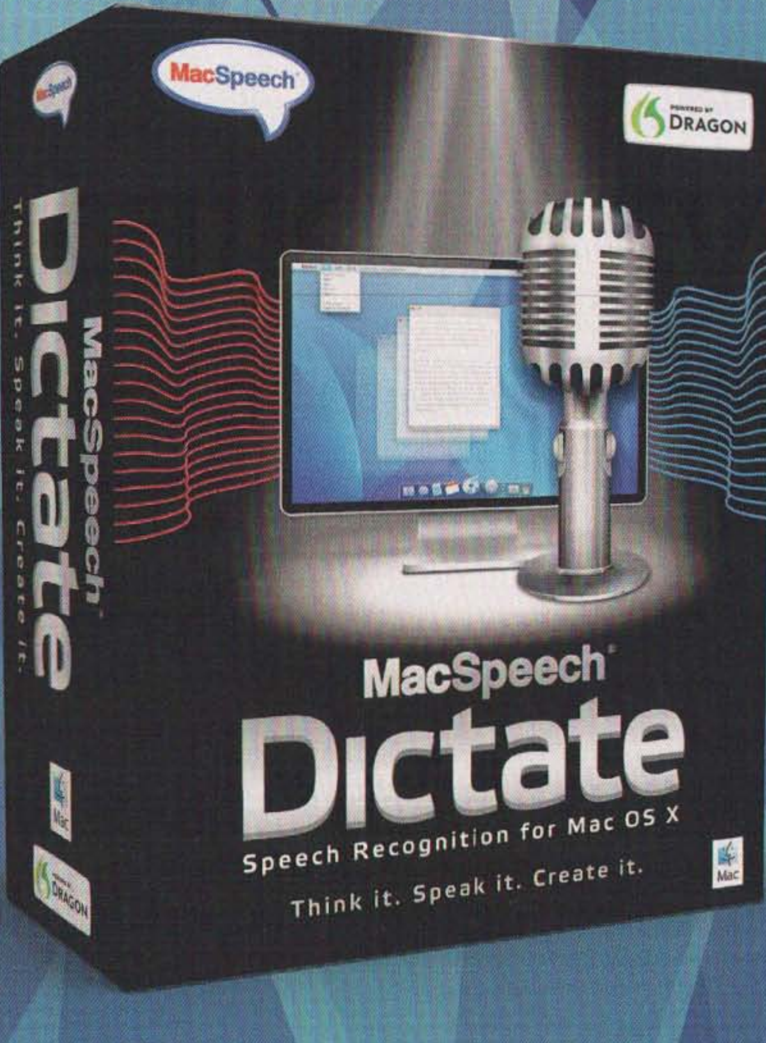
About The Author

Greg Neagle is a member of the steering committee of the Mac OS X Enterprise Project (macenterprise.org) and is a senior systems engineer at a large animation studio. Greg has been working with the Mac since 1984, and with OS X since its release. He can be reached at gregneagle@mac.com.


Advanced Security
WORKSHOPS
Go4Cast, provides workshops on advanced security topics:
• Digital Signatures and Email Encryption
• Creating a Certificate Authority on OS X
• 802.1x Security on OS X SmartCards and 2-Factor Authentication
Both Classroom and Web Based Training Available
GO 4 CAST
ASSESS, ADAPT, ACHIEVE
8452 S. Federal Highway | Port Saint Lucie, FL 34952
888-247-1616 | go4cast.com
Join Us on Facebook!

You talk. It types.

MacSpeech Dictate
Premier speech recognition for the Mac.


MacSpeech
POWERED BY DRAGON
MacSpeech Dictate
Think it. Speak it. Create it.
MacSpeech
Dictate
Speech Recognition for Mac OS X
Think it. Speak it. Create it.
Mac

www.macspeech.com

Available from MacSpeech, Apple, and other fine Macintosh retailers.
Visit the MacSpeech website for a complete retailer listing.

MacSpeech

~~Type~~. Talk!

POWERED BY DRAGON

CORESEC: SECURITY TOPICS FOR ADMINISTRATORS AND PROGRAMMERS:

Knock Knock, It's the Year 2010 Calling

An interview with Sven Gossel, CEO of Charismathics.

By Michele (Mike) Hjörleifsson

State of the Union

It's ten years into the new millennium and people are still using usernames and passwords, which are typically stuck on a Post-it note under their keyboard, as their primary form of authentication. Haven't we learned anything from the myriad of identity theft and data breaches that have been reported over the last decade? As an administrator you have options to enhance your authentication strength and this month we will discuss one potential option that may fit your organization and present an interview with the CEO of Charismathics, an organization that provides a robust two-factor solution for Mac OS X.

Multiple Factor Authentication

You may have heard the term two or three-factor authentication when folks discuss those little dongles on their key chains from RSA (http://www.rsa.com/) or VASCO (http://www.vasco.com/) , but what does multi-factor authentication really mean? It's quite simple actually. The first factor is something you know such as a username and password, and the second factor is something you have such as an RSA or VASCO token, smart card, or other authentication device. And, lLast but not least, the third factor is something that determines who you are, such as a fingerprint, retinal scan, palm scan, or other biometric feature that distinguishes you from someone else. We have all seen the movies depicting these types of authentication schemes but what is practical for your organization? Let's look at some of the types of two and three-factor authentication to inform you of your options.

Two-factor authentication requires you to have something and know something, hence the two factors. The 'know something' component is simply accomplished with a username and password. The 'have something' is where the waters get muddied a bit as "something" presents quite a few options. The most popular options in today's market are categorized into either One-Time-Password (OTP) tokens or smart cards. OTP tokens use a mathematical formula based on time and a device's serial number on the client end. A separate server is required in your infrastructure that contains the serial numbers of your devices and the secret algorithm used to calculate the time-based formula that validates the one time password presented by that device at any point in time. There is no network communication required between the device and your server, which typically confuses folks about the functionality of this type of authentication method. Let me simplify the operations. If you are in Tokyo and I am in New York and we both set our watches to Greenwich mean time (GMT) and calculate 2 + the current minute we will end up with the same result every time. Although this is a drastically simplified algorithm, this is exactly how the OTP devices function. So what is the down side to OTP devices? First, the battery in the device tends to die after a period of time. You also need a server in your infrastructure to register the devices to and integrate it with your current directory service. The server requirement can be quite costly.

In the interest of full disclosure I must profess I have worked in the public key infrastructure (PKI) world for years so I do have a slight bias toward PKI-based solutions. Smart cards are another two-factor option that leverage PKI technology. An administrator establishes a certificate authority and issues their users' identities on these cards. The users set a PIN (personal identification) number to unlock their credentials. The benefits to smart card technology as compared to OTP are quite distinct. Primarily, you don't need a separate authentication server to calculate a special algorithm that is then integrated into your authentication infrastructure. Mac OS X has the built-in capability to create a certificate authority and can be integrated into Open Directory, with some modifications, to provide authentication. There are a myriad of different smart card physical device types in the market place, from the traditional credit card type smart card to the

MoneyWorks®

Seriously Good Accounting Software for the Mac.

100% cross platform

Running natively on and seamlessly between Mac OSX 10.3.9 or later (including Leopard & Snow Leopard) and Windows 2000 or later (including XP, Vista & Windows 7)

And of course the accountants like us, they get a free copy that runs on Windows.

Learn more at www.moneyworks.ca
email: info@moneyworks.ca or call 800-818-6955

newer USB stick smart cards, like the one that Charismathics manufactures. Another significant benefit to using smart cards is that if the card is lost or stolen it is useless without the PIN number. And, it'sThey're easy to use since most people don't forget their PIN numbers. For added security, the smart card can be encoded to be non-exportable, thwarting any attempts to copy your identity from the card.

Background on Charismathics

Charismathics is a seven-year-old company based in Munich, Germany that provides PKI-centric authentication solutions to various operating system platforms, both desktop and mobile. They are credited with one of the biggest selling USB smart card with memory extension in the world, among other accolades. Their Mac OS X offering provides a small USB form factor smart card with integrated reader and one gigabyte or more of actual storage like a normal USB stick.

Interview with Sven Gossel, CEO of Charismathics

MacTech: Sven, can you tell me why most administrators are unaware of the smart card options available to their organizations in today's market?

Sven: The market as you described it has been dominated by OTP players for the last 10 years. The reason for that is simply that the PKI standard itself has not been developed in the sense that, although everybody knew the idea of the technology, implementing that technology into hardware and software, actually took a long time. Fifteen years ago when companies like Entrust or Baltimore started, they sold very large vertical solutions that only companies of more than 100,000 employees could actually afford.

MacTech: For the uninitiated, Apple touts the fact that they have built-in smart card support but most people don't understand that you actually need some middleware, like the software that Charismathics provides, to initialize and create your own cryptographic credentials, etc. Could you give a 10,000 ft. view to an uninitiated user about what your middleware brings to an Apple ecosystem?

Sven: Apple´s built-in smart card support is very project specific and is not meant to solve the need for a typical use case of individuals or small businesses. Also, For for the typical Apple user, just the middleware would only solve 50% of the issue. Users have to have a smart card and a smart card reader like our USB stick reader that puts these two things together into a simple form factor. So what we provide to the Apple world is not just the middleware software butbut also the actual device for an end-to-end solution that integrates well with Mac OS X.

MacTech: There have been several, single user-based solutions where the user credentials are created locally. What about in-network environments where your administrators are typically issuing your credentials? How does your solution work in that environment?

Sven: On top of the middleware and physical token side of the equation we do provide a management console that we call *security token configurator*. This is the application that the small enterprise requires to set up and initiate certificates, and request certificates through a managed PKI service, such as Verisign or any other certificate authority. You can even create your own certificates.

MacTech: Do you see any of the regulatory requirements in the U.S. like, the Sarbanes-Oxley, HIPPA or the PCI regulations for credit card transactions augmenting or motivating people to implement smart card solutions to achieve a two-factor solution?

Sven: Technically they do. For the Apple market you have that market that is targeting the enterprise sector. That's the one you just described. The technology does address the strong authentication required by these regulations. Our software and this technology include solutions to these regulations as well.

MacTech: The common response I get when I speak to smaller companies about implementing two-factor authentication is "Hey you know we're a small company, we don't really need that kind of technology." How would you respond to that?

Sven: I think that shows the experience people had about 5-10 years ago. Implementing PKI with its basic functionality for user authentication and digital signatures does not need to be a big project, nor does it need big money. And, if you take, say, a 10 user PKI, I'm pretty sure that we can have that solution for a 10 person company for way less than $1,000.00. So, what I'm describing here is a situation that has dramatically changed over the last 5 years where you don't need to get to the big schemes. You can have a small installation providing the same level of security for your daily use, with way less cost and less complexity. With our product, you just install the software, you follow the way through the manual and boom - you run your own PKI system. I wouldn't go so deep saying this is already easier easy enough for the home user, but we are almost there. Everybody who's a small enterprise or just doing his individual professional businesses, is able to use the software, and it's relatively simple.

MacTech: You mentioned earlier that people don't have to go out and put in big PKI infrastructures. So, could they leverage the built-in certificate authority (CA) that Mac OS X provides or some of the open source CA like EJBCA or OpenCA?

Sven: Sure, it's compatible. That's the benefit of PKI. It's not a vertical product. It's a horizontal product that provides APIs and our product is compatible with all of those APIs.

MacTech: Speaking about the exploding mobile device market, how do you see yourself positioned for things like iOS and some of the newer devices like the iPad as well as some of these higher end, more powerful computing devices that are finding their way into businesses all over the place?

Sven: We have a product for those, as well. Our software not only works on Mac OS X, obviously, we have solutions for the other bigger hardware platforms such as Windows, Linux, and Solaris. But, more importantly, more and more individuals have a smart phone today and smart phone is not only used in enterprise areas but also for the individual himself. Gartner says that in 3 or 4 years

Also available on Blackberry & Windows Mobile

Back Together Again.

Integrating calendar events and personal tasks into one easy to use App. The way it should be.

Pocket Informant

Sync is here *Pocket Informant, the iPhone's most versatile calendar/tasks Personal Information Manager (PIM) sets itself apart from the rest by providing the ability to sync to the other systems you hold dear. Now syncing with Google, Toodledo, Outlook, and Mac OS X.*

(note: Desktop Sync information available at our website)

www.pocketinformant.com | *2010 ©*

almost every phone sold, will be a smart phone. So now you have a high performance device in your hands and we are making use of it. We don not only provide the same functionality that we have on the bigger hardware platforms onto the smart phones, but we are also connecting the smart phone to the PC replacing even the need for that hardware. This product is called iEngima, and for people who would like to have that functionality, this product allows them to perform that kind of security without that smart card and smart card reader. Instead, you basically put that application into a smart phone and then connect to the computer and are essentially doing the same thing. iEngima is not yet available on the iPhone but we are planning on a release for iOS.

Three Factor

Three factor solutions have been around for the last ten years or so in several form factors including fingerprint scanners, retinal scanners and more recently capillary and facial recognition scanners. While these add an additional factor of security, many have proven impractical or lack acceptance by the user community. For instance, retinal scanners seem to scare the average user— having a red or blue light scan your eyeball just doesn't feel natural or safe though I am assured it is 100% safe. Of these solutions I have tested, read about or implemented, the most promising are capillary scanners provided by companies like Fujitsu and facial recognition scanners. Currently there aren't any options on the market for Mac OS X using theses devices. Time and testing, I am sure, will bring more awareness to these devices and additional potential security devices that utilize the third factor of authentication.

Conclusion

We are passing the first decade of the new millennium, yet most organizations are still using authentication technologies that date back to the first days of computer security. and These methods have repeatedly proven to be unreliable, easily hacked and just plain insecure. Isn't it time you examined the options available to your organization to protect your data and digital identities?

MT

About The Author

Michele (Mike) Hjörleifsson, co-author of the* Apple Training Series: Security and Mobility *courseware has been developing on the Apple platforms since the Apple][+, implementing network and remote access security technologies since the early '90s, and worked with the nation's largest corporations and government institutions, authoring white-papers, technical magazine articles and topical discussions at IETF (Internet Engineering Task Force), and other organizations on security topics, and podcasting with Apple Podcast Producer. He is currently working with companies worldwide on Apple and Security consulting projects and conducting Apple IT and Pro Apps training. Feel free to contact him at mhjorleifsson@me.com

100% Mac-compatible Web Hosting

- 99.9% Uptime
- 45-day Money Back Guarantee
- Over 4,500 Free Web Design Templates
- Free Site Builder Software
- In-house Support Available 24x7x365

Sign up at www.HostGator.com/mactech and get the first month for just a penny!

Already have a web site? We'll transfer it for free!

www.HostGator.com 866.96.GATOR

Scripting with SatImage, Part 1

Introducing the SatImage scripting addition

by José R.C. Cruz

Introduction

One of the appeals of AppleScript is its *approachable, easy to use* language syntax. It does not intimidate users with bizarre word constructs like those found in Perl. Nor does it confuse them with inconsistent syntax constructs like those that plague bash shell scripts. It is, to Apple's point of view, the scripting language for the rest of us.

Yet, some of us have to work with data of a scientific or technical nature. We may find AppleScript unable to handle such data because it lacks the means to do so. For these cases, we have to rely on the SatImage scripting addition.

Readers are expected to have a working knowledge of core AppleScript and of the Script Editor utility. The scripts featured in this article are all available from the MacTech website at:

ftp://ftp.mactech.com.

Extending AppleScript

A popular way of extending the core AppleScript language is with *scripting additions*. These specialized bundles add new verbs and nouns to the language. Each noun creates a new data object or coerces an existing object into the desired form. Each verb runs a compiled code routine, which may display an interface widget, access a Cocoa framework, or run a unique process.

All scripting additions go into the directory `ScriptingAdditions`. Where this directory is placed sets the access level to each addition. If the directory is placed in `/Library`, its additions are then available to all users. But if the directory is placed inside `~/Library`, which means a user's home directory, then only that user can access the additions therein.

The Standard Additions

One good example of a scripting addition is the **Standard Additions**, which comes with every version of MacOS X. This addition supplies many of many of the routines and services needed by a script or workflow.

Figure 1 shows the seven basic sets of verbs and nouns that make up the addition. In the first set (*brown*) are verbs that allow scripts to interact with their users. For instance, one script could display a standard Open File dialog, with which a user can select a file. Another script could show a list of scriptable applications that it can try to control. Some scripts may choose to prompt the users with a standard alert dialog. Some may choose to read their text data using a text-to-speech process.

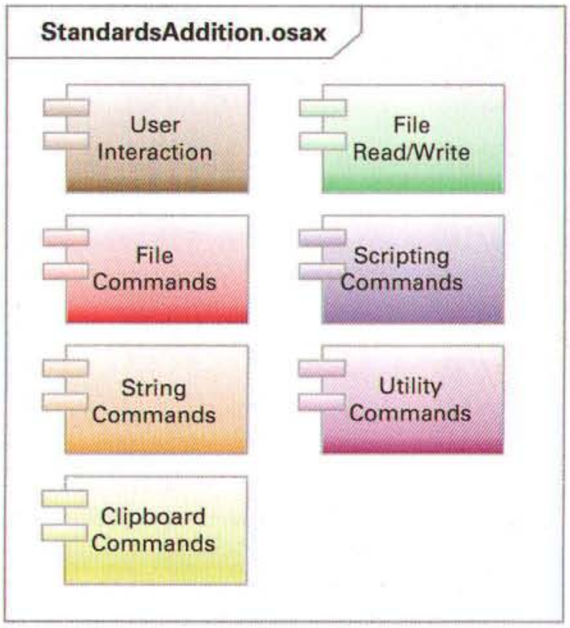


Figure 1. The Standard Additions.

In the second set (*red*) are verbs and nouns that let scripts work with files and volumes. Some scripts may need the metadata of a given file. Others may need a list of logical volumes. Some could get the absolute path of one of many standard directories. And some could even mount a specific logical or network volume.

The third set of verbs and nouns (*orange*) give scripts the ability to work with string data. Scripts can convert each string character to its unique ASCII code. They can read localized strings stored inside a `.strings` file. They can locate specific substrings inside other strings. And they can reduce a large text string into its bare essentials.

The fourth set (*yellow*) allow scripts to interact with the global system clipboard. Scripts can send data to the clipboard or read any data it held. They can even check the type of data currently present in the clipboard.

Now in the fifth set (*green*) are verbs that enable scripts to process file data. With it, scripts can open a file for access. They can read blocks of data from the file or write new data to the same file. They can also measure the file's actual size or change it to a manageable size.

With the sixth set of verbs (*blue*), scripts gain the ability to work with other scripts. They can load a script into memory and invoke its specific routines. They can also store parts of themselves into a file for later use. They can even query the core AppleScript engine for a list of available scripting components.

And with the seventh set of verbs (*purple*), scripts gain some basic system services. They can find out the current system time or any of the host system's attributes. They can adjust the current system volume or run an external POSIX script file. Plus, they get a reliable random number generator.

The SatImage Addition

For our average scripting needs, we will find the core AppleScript language and the Standard Additions more than sufficient. But suppose we are in a lab where our data comes from multiple data acquisition gear. Or we are part of a land survey team assigned to measure the distances and directions of several key landmarks. Or we work at the census bureau looking for patterns and trends in our collected data.

In these situations, we may find both AppleScript and the Standard Additions unable to process our technical and scientific data. We could use separate shell scripts to process the data, and still use AppleScript to control and coordinate those scripts. But this means we have to employ someone else to write those scripts or learn how write the scripts themselves.

Perhaps a better solution is to rely on a unique scripting addition called **SatImage**.

Overview of the addition

The SatImage addition adds five new sets of nouns and verbs to the core AppleScript language (Figure 2). The first set consists of nouns and verbs that perform faster and more intricate text operations (*brown*). The verbs themselves can use the same string object provided by AppleScript.

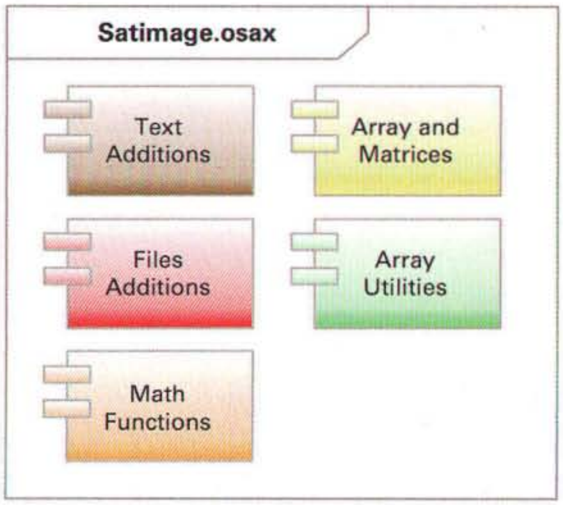


Figure 2. The SatImage addition.

In the second set (*red*) are nouns and verbs that allow scripts to interact with files, directories, and volumes. Some verbs present user dialogs with options not offered by those from Standard Additions. Some work with URL paths and supply basic server information.

The next set of verbs and nouns (*yellow*) grant scripts the ability to create and manage array and matrix objects. Arrays and matrices are packed structures of ordered real number values. They figure prominently in many technical problems, especially those that involve filters, linear programming and statistics. There is even a separate set (*green*) that sort, mask and suppress array data.

The last set of verbs (*orange*) is what makes SatImage useful to the technically minded. It is the set that supplies the common mathematical functions needed to process and analyze scientific data.

At the time of writing, the latest version of the SatImage addition is *3.5.2*. It is available separately or as part of the SmileLab IDE. To get a copy of the SmileLab installer (version 3.5.3, build 611), go to this URL.
http://www.satimage.fr/software/downloads/Smile611.pkg

To get just the SatImage scripting addition, go to this URL.
http://www.satimage.fr/software/downloads/Satimage360.pkg

We will be unable to study every aspect of the SatImage addition, for reasons of length. But we will focus on those aspects that are unique and those that most scripts will find useful.

Working with Text

Now core AppleScript has only a barebones text search feature. It is incapable of replacing portions of text or of using intricate string patterns. Furthermore, it lacks any conversion services, relying instead on users to supply them as script routines. Thus, when large amounts of text are involved, core AppleScript is either too slow or too limiting.

This is where SatImage can help.

Searching the text

To search for a string within a given text, use the verb `find text` (Figure 3). The verb takes two arguments: the target text of the search and the string to be searched. It starts the search from the left of the text and assigns the leftmost character with a value of zero. If the search is successful, the verb returns its results as a `matchrecord` object.

```
            ⎧ search-string
find text   ⎨
            ⎩ in source-text
```

Figure 3. The find text verb.

Assume our target text is as follows.

```
set tTxt to "The big brown fox jumps over the lazy dog."
```

To search for the word `"fox"`, use the `find text` verb as follows.

```
find text "fox" in tTxt
— returns {matchPos:14, matchLen:3, matchResult:"fox"}
```

The returned `matchrecord` object divides the search results into three fields. The field `matchPos` gives the position of the first match. The field `matchLen` gives the length of the match, and the field `matchResult` gives the matching string itself. In most cases, the string in `matchResult` will be equal to the search string itself.

Suppose the target text does not have the search string. In this case, the `find text` verb will return an error code of `-2763`.

```
find text "cat" in tStr
— returns -2763:"No result was returned from some part of this
expression."
```

This error means that the verb did not find the string `"cat"` anywhere in the target text.

Two groups of options control the action of the `find text` verb. One group controls the *range* and *start* of the search (Figure 4.a). The other group controls the *search behavior* (Figure 4.b). Assume again the target text featured earlier. To search for the word `"lazy"` after finding the word `"fox"`, pass the new start position to the option `starting at`.

```
find text "fox" in tTxt
— returns {matchPos:14, matchLen:3, matchResult:"fox"}

set tPos to matchPos of result
find text "lazy" in tTxt starting at tPos
— returns {matchPos:33, matchLen:4, matchResult:"lazy"}
```

To search for every instance of the word `"the"`, set the `case sensitive` option to `false`, then the `all occurrences` option to `true`.

```
find text "the" in "The big brown fox jumps over the lazy
dog" ¬
    all occurrences true case sensitive false
— returns
— {{matchPos:0, matchLen:3, matchResult:"The"},¬
    {matchPos:29, matchLen:3, matchResult:"the"}}
```

Here the verb returns a *list* of `matchrecord` objects instead of a single object. To get only the matching strings, set the `string results` option to `true`.

```
find text "the" in "The big brown fox jumps over the lazy
dog" ¬
    all occurrences true case sensitive false string results
true
— returns {"The", "the"}
```

Here again the verb returns the matching strings as a list.

```
            ⎧ search-string   ⎡ starting at search-position
find text   ⎨                 ⎢
            ⎩ in source-text  ⎣ for search-range
```

(a) range and position

```
                              ⎡ case sensitive settings-flag
            ⎧ search-string   ⎢ whole word settings-flag
find text   ⎨                 ⎢ all occurrences settings-flag
            ⎩ in source-text  ⎣ string result settings-flag
```

(b) search behavior

Figure 4. The find text options.

Changing the text

To change parts of a given text, use the verb `change` (Figure 5). This one takes three arguments: the target text, the search string, and the replacement string. When the verb is successful, it returns the *modified text* as the result. Otherwise, it returns the *original, unaltered text*.

```
         ⎧ search-string
change   ⎨ into replacement-string
         ⎩ in source-text
```

Figure 5. The change verb.

Consider this script snippet:

```
set tStr to "The big brown fox jumps over the lazy dog"
change "fox" into "cat" in tStr
— returns "The big brown cat jumps over the lazy dog"
```

Here, the verb looks for the word `"fox"` in the target text. It then replaces the word with `"cat"` and returns the altered string. But the original text held by the local `tStr` remains the same.

The `change` verb also has two groups of options (Figure 6). Again, one group controls the start of the search, the other group the search behavior.

An Awesome Email Marketing Tool for MACTECH READERS

Benchmark Email is the standard in permission-based email marketing

- ✔ List Management
- ✔ Free Newsletter Archive
- ✔ Easy to Use Drag-n-Drop Email Editor
- ✔ Powerful Personalization
- ✔ Image Gallery
- ✔ Visual Graphs for Open & Clickthrough Tracking
- ✔ Spam & Spell Checkers
- ✔ Creative & Compelling HTML Templates
- ✔ Upload Your own Template
- ✔ Easy Video Integration

benchmark email
Email Marketing Services for your Business
Trusted by 65,000 Small Businesses & Users
FREE 30-Day Trial

MACTECH users get a 10% Lifetime Discount with this promo code 123923

Most email marketing services charge more money for less product. Not us. Benchmark Email's sophisticated suite of email marketing features lets you grow your list, send campaigns, track your data and even take online polls for an affordable price.

Plans starting at only $9.99 per month

Sign up for a FREE 30 Day Trial Today!

www.BenchmarkEmail.com

benchmark email

800.430.4095

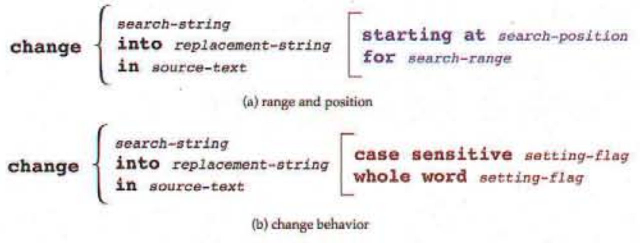


Figure 6. The change options.

Consider the following script snippet:

```
set tStr to "Peter piper picked a peck of pickled peppers."
change "pe" into "li" in tStr starting at 10
— returns "Peter piper picked a lick of pickled liplirs."
```

Here, the `change` verb searches for every instance of the string `"pe"`, which it then replaces with the string `"li"`. Since it starts the search at character 10, the verb skips over the word `"piper"`. Now consider this snippet.

```
set tStr to "Peter piper picked a peck of pickled peppers."
change "pe" into "li" in tStr starting at 10 for 20
— returns "Peter piper picked a lick of pickled peppers."
```

Here too, the verb starts its search for `"pe"` at character 10. But it ends the search after it reaches character 20. As a result, only the word `"peck"` gets changed in the process.

Consider now this script snippet:

```
set tStr to "She sells sea shells on the seashore."
change "sea" into "snail" in tStr
— returns "She sells snail shells on the snailshore."
change "sea" into "snail" in tStr with whole word
— returns "She sells snail shells on the seashore."
```

In the first change line, the script replaces every instance of the string `"sea"` to `"snail"`. But in the second change line, the script only replaces the word `"sea"`, which is an *exact match* of the search string.

Patterns in text

Both the `find text` and `change` verbs can use *regular expressions* in the place of an explict search string (Figure 7). To use a regex pattern, pass the pattern as a string and pass a `true` to the option `regexp`. To demonstrate, this code snippet finds all occurences of the word `"the"`, regardless whether or not the word itself is capitalized.

```
set tStr to "The big brown fox jumps over the lazy dog"
find text "[Tt]he" in tStr regexp true all occurrences true
— returns
— {{matchPos:0, matchLen:3, matchResult:"The"}, {matchPos:29,
matchLen:3, matchResult:"the"}}
```

Now if we pass a `false` to the `regexp` option by mistake, the `find text` verb will treat our regex pattern literally. Then we get the error code `-2763` as the search result.

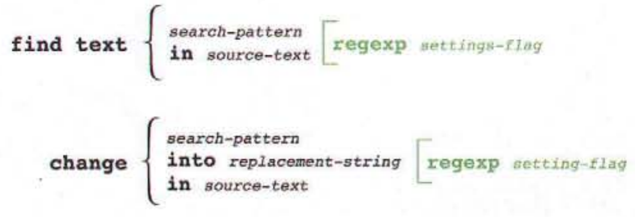


Figure 7. The regexp option.

Next, this code snippet replaces the string `"pe"` with `"ler"`, but only when `"pe"` is followed by a `"c"` or an `"r"`. Thus the word `"piper"` turns into `"piler"`, `"peck"` into `"lerk"`, and `"peppers"` into `"peplers"`. As expected, the first substring `"pe"` in the word `"peppers"` remains unchanged.

```
set tStr to "Peter piper picked a peck of pickled peppers"
change "pe[cr]" into "ler" in tStr regexp true
— returns "Peter piler picked a lerk of pickled peplers"
```

Again, if the regexp option gets a value of true, the `change` verb will treat the pattern literally and the desired effect will not happen.

Back references are allowed, as shown by the code snippet below. Here too, the `change` verb replaces string `"pe"` with the string `"le"`. But this time, the verb restores the letter after the `"pe"` string via the `'\1'` back reference. Thus, the word `"piper"` becomes `"piler"`, `"peck"` becomes `"leck"` and `"peppers"` becomes `"peplers"`.

```
set tStr to "Peter piper picked a peck of pickled peppers"
change "pe([cr])" into "le\\1" in tStr with regexp
— returns "Peter piler picked a leck of pickled peplers"
```

Note the `'\'` token appears twice—this is to force AppleScript to treat the token literally.

Extracts of text

To extract portions of the target text, use the `extract string` verb (Figure 8). The verb takes three input arguments: the target text, and the start and range of extraction. If we leave out the start and range, the verb gives the original text as its result.

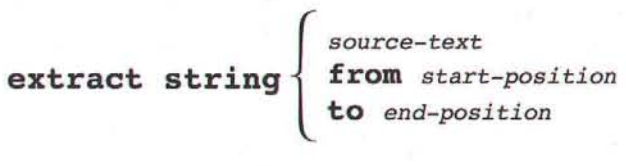


Figure 8. The extract string verb.

There are several ways to use the `extract string` verb. For instance, to extract the *first ten characters* of the text, pass the value 10 to the `to` setting. Leave the `from` setting alone.

```
set tStr to "The big brown fox jumps over the lazy dog"
extract string tStr to 10
```

```
— returns "The big br"
```

If this is done in core AppleScript, the snippet will appear as follows.

```
(characters 1 thru 10 of tStr) as string
— also returns "The big br"
```

Note the simplicity of the `extract string` line as opposed to the verbosity of the core AppleScript line.

To extract just the right portion of the source text, *starting at the 10th character*, pass the value 10 to the `from` setting. This time, leave the `to` setting alone.

```
extract string tStr from 10
— returns "rown fox jumps over the lazy dog"
```

To extract the *last ten characters* of the text, pass the value -10 to the `from` setting.

```
extract string tStr from -10
— returns "e lazy dog"
```

Then to extract the left portion of the text, *minus the last nine characters*, pass -10 to the `to` setting.

```
extract string tStr to -10
— returns "The big brown fox jumps over the"
```

Finally, to extract from the middle of the text, use both `start` and `to` options as follows.

```
extract string tStr from 10 to 15
— returns "rown f"
```

In this example, the verb starts its extraction at the 10th character and ends at the 15th character. The resulting string has a total length of six characters, and it includes both 10th and 15 characters.

Conversions in text

Several verbs convert the text from one form to another. The verb `uppercase` shifts every letter in the target text to upper case. Conversely, the verb `lowercase` downshifts the same letters to lower case.

```
set tStr to "The big brown fox jumps over the lazy dog"
uppercase tStr
— returns "THE BIG BROWN FOX JUMPS OVER THE LAZY DOG"
lowercase tStr
— returns "the big brown fox jumps over the lazy dog"
```

Both verbs use the target text as their sole input. Both apply the change wholesale. But we can use the `extract string` verb to isolate the change within a specific range.

The verb `convert to Windows` remaps the target text to the standard Windows character set. And the verb `convert to Mac` performs the opposite action. Again, both verbs take the target text as their sole input. They do not affect letters and numbers common to both platforms.

```
set tMac to "aeiou bcdfghjklmnpqrstvwxyz 0123456789"
set tWin to convert to Windows tMac
— returns "aeiou bcdfghjklmnpqrstvwxyz 0123456789"
convert to Mac tWin
— returns "aeiou bcdfghjklmnpqrstvwxyz 0123456789"
```

But they do affect those meta-symbols that differ between those two platforms.

```
set tMac to "• áéíóú àèìòù âêîôû üåæñ€¢∞§"
set tWin to convert to Windows tMac
— returns "ï ·ÈÌÛ ‡ËÏÚ˜ ‚ÍÓÛ˚ ¸ÂÊÒÄ¢?ß"
convert to Mac tWin
— returns "• áéíóú àèìòù âêîôû üåæñ€¢?§"
```

The verb `encode entities` looks for characters that have a corresponding XML entity. It then replaces each character with its entity. The verb `resolve entities` reverses the process.

```
set tStr to "&<>\"'"
set tXML to encode entities tStr
— returns "&amp;&lt;&gt;&quot;&apos;"
resolve entities tXML
— returns "&<>\"'"
```

Right now, SatImage recognizes the five character entities defined by the XML standard. But future versions of SatImage may support other entities such as those defined by SGML and HTML.

The verb `escapeURL` looks for characters that are not valid for URL strings. It then replaces each characters with its *hex code* preceded by a `'%'` token. The verb `unescapeURL` does the opposite.

```
set tStr to "Érase una noche triste"
escapeURL tStr
— returns "%C3%89rase%20una%20noche%20triste"
set tStr to "%C3%89rase%20una%20noche%20triste"
unescapeURL tStr
— returns "Érase una noche triste"
```

Use these verbs when processing text that has spaces or accented letters.

Finally, the verb `format` takes a real number value and converts it into a preformatted string (Figure 9). It takes two arguments: the real value and the format string.

```
format { real-value
       { into format-string
```

Figure 9. The format verb.

The format string can have four possible tokens. The token `'#'` marks the location of a digit. If the digit is a leading zero, the token resolves itself into a nil character.

```
set tNum to 012345
format tNum into "######"
— returns "12345"
```

The token `'0'` also marks the location of a digit. But it preserves any leading zeros present in the number.

```
set tNum to 012345
format tNum into "000000"
— returns "012345"
```

Next, the token `'.'` marks the location of the decimal point, if one is needed. Without this token, the `format` verb truncates the real value, leaving only its integer portion.

```
set tNum to 1.23456789E+4
format tNum into "######.######"
— returns "12345.6789"
format tNum into "######"
— returns "12345"
```

And the token `'%'` multiplies the real value by 100 before rendering it into a string. Then it appends itself at the end of the string.

```
set tNum to 0.12345
format tNum into "######.######%"
— returns "12.345%"
set tNum to 12.345
format tNum into "######.######%"
— returns "1234.5%"
```

The format string also allows words and phrases to appear before or after its tokens. These words and phrases then appear unaltered on the final string. Make sure none of the words use any of the format tokens.

```
set tNum to 12345
format tNum into "'The real number is '######.######"
— returns "The real number is 12345"
```

Interacting with Files and Users

Sometimes, scripts need access to a specific file or directory. Sometimes, they need users to point them to the correct file or directory. Such interactions happen because scripts need data for processing, or they have data that must be stored.

The Standard Additions has its own rich set of verbs and nouns that supply scripts the above abilities. Some of its verbs even allow some basic customizations. But for more precise and flexible interactions, we need the aid of the SatImage addition.

Asking for a file

In order for users to choose a file, the script must present them with an Open File dialog. This is done with the SatImage verb `navchoose file` (Figure 10). The verb has six optional settings, each changing a specific aspect of the dialog. The dialog itself appears as a movable modal and uses a columnar form by default. It lets users select multiple files, and it points to the last visited directory. Clicking the **Cancel** button returns a user-cancelled event, which also terminates the script. Clicking the **Open** button returns the selected file path as a list of file objects.

```
                 with prompt prompt-string
                 starting at file/folder-alias
navchoose file   multiple files settings-flag
                 show packages settings-flag
                 open packages settings-flag
                 of extension suffix-list
```

Figure 10. The navchoose file verb.

To display a barebones Open File dialog, simply use the verb as is.

```
navchoose file
— returns {file "Darwin:SqliteQuery.app:", file "Darwin:zire_vcal"}
```

BMS

THE LAW OFFICE OF
BRADLEY M. SNIDERMAN

Helping clients with their software legal issues.

- **Trademark and Copyright Registration**
- **Trade Secret Protection**
- **Licensing and Non Disclosure Agreements**
- **Assist with Software Audits**

I am an attorney practicing in Intellectual Property, Business Entity Formations, Corporate, Commercial and E-commerce Law.

Please give me a call or an e-mail. Reasonable fees.

23679 Calabasas Rd. #558 • Calabasas, CA 91302
PHONE 818-706-0631 FAX 818-706-0651 EMAIL brad@sniderman.com

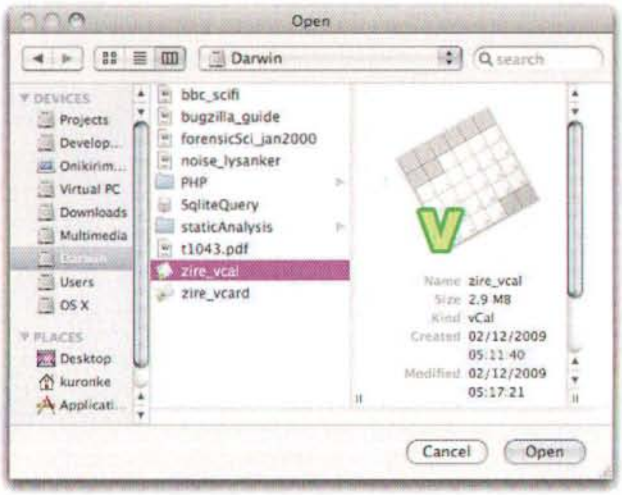


Figure 11. The Open File dialog.

To give the dialog a prompt message, pass the message string to the `with prompt` option. If the string is longer than the dialog's width, the dialog will truncate the rest of the prompt.

```
— this prompt appears normally
navchoose file with prompt "Select an existing file"
— this prompt gets truncated
navchoose file with prompt "Select a file whose contents you
want to secure with your custom cipher. But do not select an
application or package."
```

To change the default directory, pass the directory to the `starting at` option. Make sure to render the diretory path as an alias object.

```
set tPth to path to home folder
navchoose file with prompt "Select an existing file"
starting at tPth
```

To restrict selection to one file, set the `multiple files` option to `false`. The verb will still return the selected file as a list.

```
navchoose file with prompt "Select an existing file"
multiple files false
— returns {file "Darwin:zire_vcal"}
```

To exclude bundles from the selection, pass a `false` to the `show packages` option. Bundles will then appear as *disabled items* on the Open File dialog.

```
navchoose file with prompt "Select a package" show packages
false
```

On the other hand, to treat bundles as valid directories, pass a `true` to the `open packages` option. The Open File dialog will then allow users to browse and select items inside a bundle. Installer packages, however, will still appear as *distinct files*.

```
navchoose file with prompt "Select a package item" open
packages true
```

Asking for a directory

Just as the Open File dialog prompts users for a file, the Open Folder dialog (Figure 12) prompts them for a directory. The dialog also appears as a movable modal and points to the last visited directory. Any file items in the directory appear disabled, preventing their selection. Users, however, can select two or more directories. Clicking the **New Folder** button starts the process of adding a new subdirectory to the current directory. Clicking the **Choose** button returns the path to the selected directory as a file object. And clicking the **Cancel** button returns a user-cancelled event.

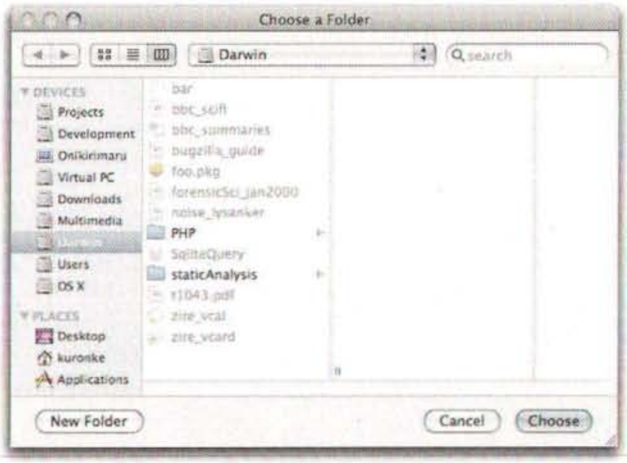


Figure 12. The Open Folder dialog.

The SatImage verb that supplies this dialog is the `navchoose folder` verb (Figure 13). This verb uses four of the options that the `navchoose file` verb also uses. So to display a barebones dialog, use the verb without any options.

```
navchoose folder
— returns {file "Darwin:PHP:", file "Darwin:staticAnalysis:"}
```

```
                    with prompt prompt-string
navchoose folder    starting at file/folder-alias
                    multiple files settings-flag
                    open packages settings-flag
```

Figure 13. The navchoose folder verb.

To give the dialog a prompt message, pass the message string to the `with prompt` option.

```
navchoose folder with prompt "Select an existing directory"
```

To use a different starting directory, pass the directory path to the `starting at` option.

```
set tPth to path to home folder
navchoose folder starting at tPth
```

To restrict selection to one directory at a time, pass a `false` to the `multiple files` option.

```
navchoose folder multiple files false
```

And to treat bundles as valid directories, pass a `true` to the `open packages` option.

```
navchoose folder with open packages
```

Asking to save

When scripts have data they need to save, they should signal their intent with a Save File dialog (Figure 14). Here too, the dialog points itself to the last visited directory. It offers a default name of `"untitled"` for the output file. The dialog appears minimized at first, but clicking its down-arrow button (next to the **Save As** field) switches the dialog to its browser mode. Clicking the **Cancel** button returns a user-cancelled event. Clicking the **Save** button returns the path to the output file as a file object. This action, however, does not create the blank file.

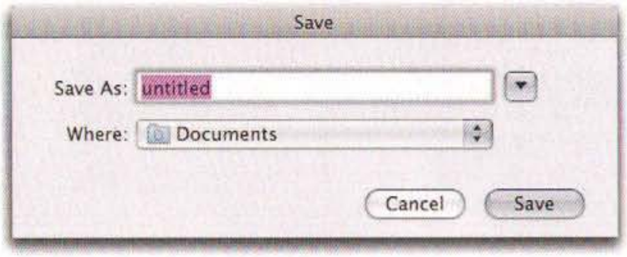
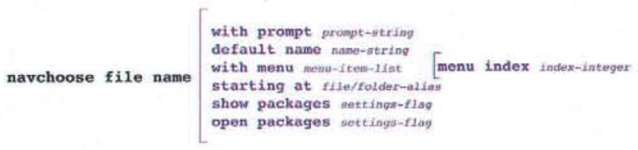


Figure 14. The Save File dialog.

The SatImage verb `navchoose file name` (Figure 15) handles the display of the Save File dialog. The verb shares some of the options as the previous two verbs—plus, it adds its own. So to display a barebones dialog, just use the verb by itself.

```
navchoose file name
- returns {file "Darwin:untitled"}
```



```
navchoose file name  with prompt prompt-string
                     default name name-string
                     with menu menu-item-list   [menu index index-integer
                     starting at file/folder-alias
                     show packages settings-flag
                     open packages settings-flag
```

Figure 15. The navchoose file name verb.

To give it a prompt message, pass the message string to the `with prompt` option.

```
navchoose file name with prompt "Save the document as"
```

To use a different starting directory, pass the directory path, again as an alias object, to the `starting at` option.

```
set tPth to path to documents folder
navchoose file name starting at tPth
```

To offer a different default name, pass the name string to the option `default name`.

```
navchoose file name default name "foobar"
```

To offer a list of output format, pass the list to the option `with menu`. This adds a pop-up menu labeled **Format** near the bottom of the Save File dialog. Then clicking the **Save** button returns a two-field record. The field `path` holds the path to the output file. The field `menu item` holds the chosen format.

```
navchoose file name with menu {"Vigenere file", "e-book
bundle"}
- returns
- {path:file "Users:Home:kuronke:Documents:untitled", menu
item:1}
```

Now the **Format** pop-up menu offers the first menu item as the default format. To change the default, pass the format's index to the option `menu index`.

```
navchoose file name with menu {"Vigenere file", "e-book
bundle"} ¬
    menu index 2
```

Making a query

With a path on hand, a script can use the SatImage addition to query the item at said path. For instance, to query a given directory, use the SatImage verb `list files`. This verb takes one argument, which is the directory path as an alias object. And it returns the query as a list of file objects.

```
set tPth to path to scripting additions
list files tPth
(*
returns
    {file "OS X:System:Library:ScriptingAdditions:ColorSyncScripting.app:"
    , file "OS X:System:Library:ScriptingAdditions:Digital Hub
Scripting.osax:"
    , file "OS X:System:Library:ScriptingAdditions:StandardAdditions.osax:"
    , file "OS X:System:Library:ScriptingAdditions:URL Access
Scripting.app:"}
*)
```

The list files verb has three optional arguments. To include invisible files to the query, pass a `true` to the `invisibles` option.

```
list files tPth invisibles true
```

To restrict the query to a specific file group, pass a file suffix to the option `of extension`.

```
list files tPth of extension "txt"
```

To treat subdirectories as valid file items, pass a `false` to the option `recursively`.

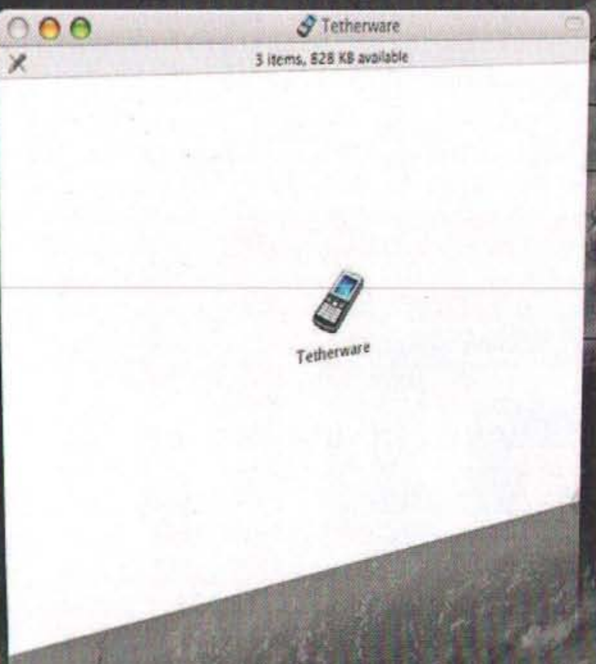
```
list files tPth recursively false
```

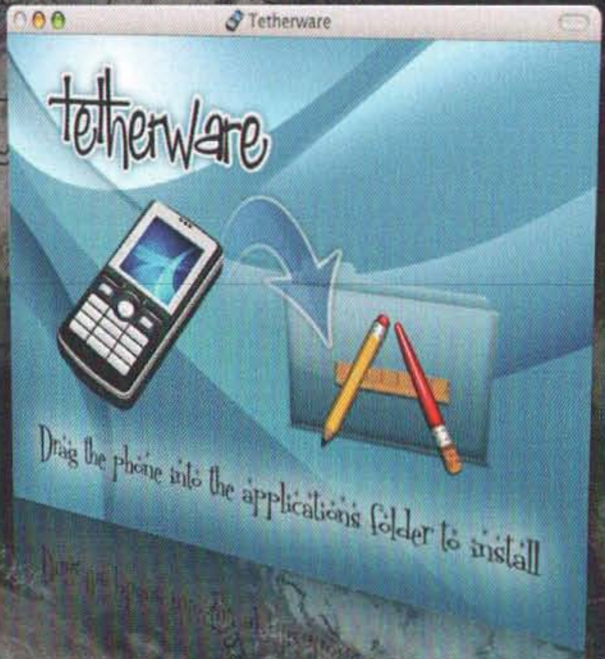
To query a file's metadata, use the SatImage verb `URL info for`. This verb takes a single argument, which is the path to the given file. The path can be an alias object or it can be a URL. The verb then returns the query as a record object, the contents of which differs with each file path.

Consider this sample snippet. It gives the metadata of the Standard Additions bundle that lies on the local boot volume:

```
set tPth to path to scripting additions
set tPth to (tPth as string) & "StandardAdditions.osax"
set tPth to tPth as alias
URL info for tPth
(*
returns
    {name:"StandardAdditions.osax"
    , creation date:date "Sunday, September 23, 2007 22:31:34"
    , modification date:date "Sunday, September 23, 2007 22:31:34"
    , size:1229920, folder:true, alias:false, package folder:true
    , visible:true, extension hidden:false, name extension:"osax"
    , ... truncated for length
    }
*)
```

Does your DMG look damaged?





DMG built WITHOUT FileStorm in OS X 10.6 displayed on 10.5 or earlier

DMG built WITH FileStorm in OS X 10.6 displayed on 10.5 or earlier

FileStorm from MindVision Software.

For when your image is hurting your image.

Purchase and download now
www.mindvision.com/mactech

Now look at this snippet. This one gives the metadata of an image file from a Wikipaedia web page.

```
set tURL to "http://en.wikipedia.org/wiki/File:Area88Shin.jpg"
URL info for tURL
(*
    returns
        {scheme:"http", host:"en.wikipedia.org"
        , unix path:"/wiki/File:Area88Shin.jpg
        , path:"/wiki/File:Area88Shin.jpg", name:"File:Area88Shin.jpg"
        , name extension:"jpg", type identifier:"public.jpeg"
        , file type:"", file creator:"", folder:false, package folder:false,
        alias:false
        }
*)
```

Note how both query results give a different set of fields.

Making a backup

Unique to the SatImage addition is the verb **backup** (Figure 16). This verb allow scripts to do file-level backups from one logical volume to another. The backup itself runs as an atomic process; that is, it runs to completion, without interruption.

```
backup source-alias         level backup-action
       onto store-alias     after file-dates
                            recursively settings-flag
```

Figure 16. The backup verb.

The **backup** verb takes two input arguments: the backup *source* and the *store*. The source may be a single file or it may be a directory. The store must be an existing readable directory. Both arguments are expressed as alias objects.

Suppose we want our script to backup the contents of our **Documents** directory onto the logical volume named **Archive**. To try a backup, use the verb as follows:

```
set tSrc to path to documents folder
set tDst to "Archive:" as alias
backup tSrc onto tDst
```

This snippet, however, only lists the files and directories that will be backed up from the **Documents** directory. It will not do the actual backup.

So to actually run the backup, set the **level** option to **2**. Now the **backup** verb copies each item in the **Documents** directory to the **Archive** volume. If the item is a subdirectory, the verb copies any contents inside that subdirectory. If the item is an alias, the verb copies the file or directory pointed by the alias, but not the alias itself:

```
backup tSrc onto tDst level 2
```

Here too, the verb lists each item it copies to the backup store. To disable the list and perhaps increase backup speed, set the **level** option to **1**:

```
backup tSrc onto tDst level 1
```

To backup just the directories, not their contents, pass a **false** to the option **recursively**:

```
backup tSrc onto tDst level 2 recursively false
```

To backup only items of a specific date, pass a date object to the option **after**. The verb then copies only the files and directories whose contents have *changed* since the given date:

```
set tClk to date "Saturday, May 15, 2010 00:00:00"
backup tSrc onto tDst after tClk
```

Concluding Remarks

Thus, we learned how SatImage can aid our scripts interact with users and files. We also learned the many ways it can process our text data. Next time we visit SatImage, we will study how it creates and processes arrays and matrices, data constructs that form many scientific data. We will also explore its rich set of mathematical functions and algorithms.

Be sure to visit the SmileLab website for more information about SatImage or its companion products. The website URL is http://www.satimage.fr/software/en/index.html.

Bibliography and References

Perry, Bruce W. "Standard Scripting Additions". *AppleScript in a Nutshell*. Sebastopol, CA:O'Reilly Media. 2001 June. pp. 437-466.

SatImage. "Dictionary of SatImage." Internet: http://www.satimage.fr/software/en/dictionaries/dict_satimage.html, 2008. [2010 Jul].

SatImage. "Text Commands." Internet: http://www.satimage.fr/software/en/smile/text/index.html, 2008. [2010 May].

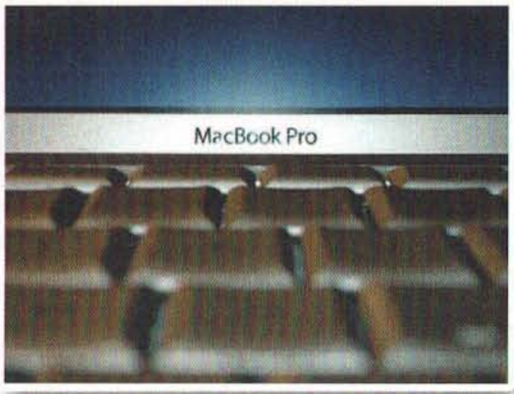
MT

About The Author

JC is a freelance engineering writer from North Vancouver, British Columbia. He frequently contributes articles to MacTech and REALbasic Developer. He also wrote for the now defunct Python Magazine, and is now working on a database e-Book. When away from the writing pile, JC spends quality time with his charming foster nephew. He can be reached at anarakisware-at-gmail-dot-com.

Short Term Computer Needs?

- Latest Apple Technology
- Legacy Macs
- Windows Computers



- Rent One or Hundreds
- Audiovisual Gear
- Projectors, Plasmas, LCDs
- B&W & Color Copiers

Mac Rentals
Event Technology Services

- Name Brands
- Nationwide Delivery
- Professional Setup
- Onsite Technicians
- Same Day Service

- Compatibility & Compliance Testing
- Short Term Projects, Production Needs
- Trade Shows, Conventions, Corporate Events
- Trainings Sessions, Classrooms
- Fortune 100 and Educational P.O.'s Accepted!

Toll Free
800-756-6227
Outside US/Canada: 858-454-8535

For Online Quote Request, Visit:

www.madcomputing.com

Largest Apple Rental Inventory in the U.S.!

Creating a Dashboard Widget that Supports Localization

by Mihalis Tsoukalos

Introduction

As your potential Widget audience may be the whole world, it will be worthwhile to think about localizing the Widgets that you create. Localization is the process of including more than one language for every message or text that appears inside your own Widgets.

This article shows you a Widget that implements the following simple task: it just displays the spelling of numbers 1, 2 and 3, supporting English, French and Greek localizations.

How Mac OS X handles localization

For most applications on Mac OS X, localized resources such as images, strings, and nib files can be found within the application's bundle, inside the **./Contents/Resources/** directory. Each supported language has its own separate directory named after the language whose resources it holds. Its name and location within the bundle are strict, as Mac OS X is expecting it to be in the right place if a particular localization is requested. These folders are called language project directories and always end with the **.lproj** extension.

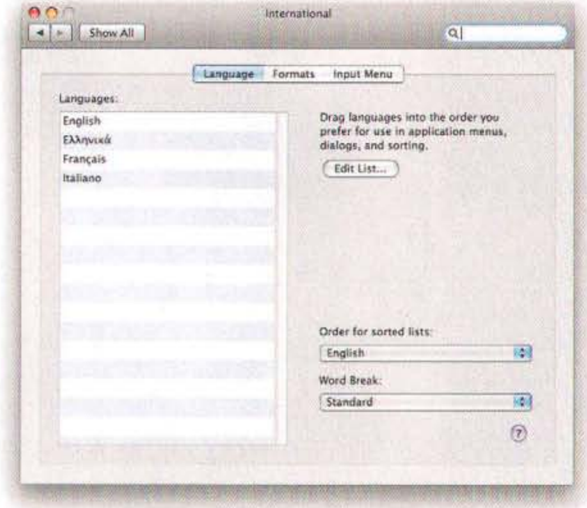
Some examples are: *en.lproj* for the English language, *el.lproj* for the Greek language, *de.lproj* for the German language, *fr.lproj* for the French language, *ja.lproj* for the Japanese language, *ko.lproj* for the Korean language, etc.When an application is launched, the executable file asks Mac OS X for certain localized resources. When this happens, Mac OS X looks for a language project within the application's bundle that corresponds with the first entry in the language precedence list, as set by the user in System Preferences (a language precedence example is shown in Figure 1).

If no language project for the preferred language is available then Mac OS X looks for a language project related to the next language in the precedence list, and so on. Note that this process is generally automatic, which means that the application is not involved in the actual searching of the language projects; it basically requests resources and Mac OS X makes them available.

How Dashboard handles localization

More or less, Dashboard works the same way as Mac OS X itself in respect to localization. The only noteworthy difference is that each language project directory needs to be located at the root level of your Dashboard Widget.

I will now present you the most important files and directories of the Local Widget. Later on I will provide the full list of the Widget's files and directories.



Languages:
English
Ελληνικά
Français
Italiano

Drag languages into the order you prefer for use in application menus, dialogs, and sorting.

Order for sorted lists: English

Word Break: Standard

Figure 1: Language precedence list in System Preferences

The Info.plist file

The contents of the Info.plist file in text format are as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST
1.0//EN"
            "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
    <plist version="1.0">
    <dict>
```

MACNEWS™

News and information for Apple users.

www.macnews.com

```
        <key>CFBundleDisplayName</key>
        <string>Local</string>
        <key>CFBundleIdentifier</key>
        <string>com.mtsouk.widget.local</string>
        <key>CFBundleName</key>
        <string>Local Widget</string>
        <key>CFBundleShortVersionString</key>
        <string>1.0</string>
        <key>CFBundleVersion</key>
        <string>1.0</string>
        <key>CloseBoxInsetX</key>
        <integer>16</integer>
        <key>CloseBoxInsetY</key>
        <integer>17</integer>
        <key>MainHTML</key>
        <string>Local.html</string>
    </dict>
    </plist>
```

There is nothing special here—this is a very standard Info.plist file.

The Local.html file

The contents of the Local.html HTML file are as follows:

```
<!—

    File: Local.html
    Programmer: Mihalis Tsoukalos
—>

    <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
              "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
    <meta http-equiv="Content-Type" content="text/html;
         charset=ISO-8859-1"/>

    <!— The style sheet should be kept in a separate file;
        it contains the design for the widget.
        Note that each language project directory in
        this widget has its own Local.css file.
        This is because each localization of this widget
        can have a different design, based on the language.
        Dashboard will automatically provide the widget
        the correct CSS file based on the user's
        language preference.
     —>
    <style type="text/css">
        @import "Local.css";
    </style>

    <!— The JavaScript file contains the logic
        needed for this widget, including the
        localization retrieval and logic.
    —>

    <script type='text/javascript' src='Local.js'
charset='utf-8'/>

    <!— A localizedStrings.js file is in every language
        project directory in this widget; the version
        opened here depends on the user's language
        preferences as chosen in System Preferences.
     —>
    <script type='text/javascript' src='localizedStrings.js'
        charset='utf-8'/>

    </head>
```

```
    <!— setup() attempts to localize the body text —>
    <body onload="setup();">

            <img src="Default.png"/>

            <div id="NumberText">1
            <br />2
            <br />3
            </div>

            <!- the language currently being used, in English
->
            <div id="language">Default</div>
    </body>
    </html>
```

The 1, 2 and 3 numbers found in the Local.html file are going to be used afterwards in order to find their respective localized strings.

The Local.js file

As this Widget is pretty simple, the Local.js JavaScript file is also simple. Its contents are as follows:

```
    /*

    File: Local.js
    Programmer: Mihalis Tsoukalos

    */


    // getLocalizedString() pulls a string out an array
    // named localizedStrings. Each language project
    // directory in this widget contains a file named
    // "localizedStrings.js", which, in turn, contains
    // an array called localizedStrings.
    // This method queries the array of the file of whichever language
    // has highest precidence, according to the International pane
    // of System Preferences.
    function getLocalizedString(key)
    {
        try
        {
            var ret = localizedStrings[key];
            if (ret == undefined)
            {
                ret = key;
            }
            return ret;
        } catch (ex) {}

        return key;
    }

    // setup() is called when the body of the widget is loaded.
    // This function places the localized
    // strings into the widget.
    function setup()
    {

    // Get the Localized strings for
    // EACH one of the three numbers
        l1 = getLocalizedString('1');
        l2 = getLocalizedString('2');
        l3 = getLocalizedString('3');

    // Format the HTML output
        LocalizedOutput = l1 + "<br>" + l2 + "<br>" + l3;

        document.getElementById('NumberText').innerHTML =
LocalizedOutput;
```

```
        document.getElementById('language').innerText =
getLocalizedString('Default');
    }
```

This file is explained in more detail in the section, "Explaining The Technique."

The Local.css file

File Local.css contains the following data:

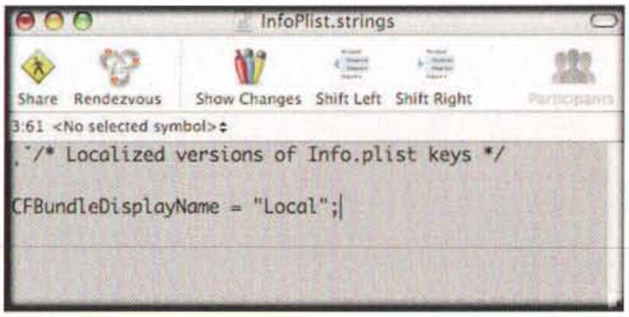
```
/*

File: Local.css
Programmer: Mihalis Tsoukalos

*/

body
{
    margin: 0;
}

#NumberText
{
    font: 15px "Times";
    font-weight: bold;
    color: white;
    text-align: center;
    position: absolute;
    top: 24px;
    left: 10px;
    width: 200px;
}

#language
{
    font: 10px "Times";
    color: white;
    text-align: center;
    position: absolute;
    top: 90px;
    left: 18px;
    width: 180px;
}
```

As you can see, there is nothing special here. After all, Local is a pretty simple Dashboard Widget.

The en.lproj directory contents

The **en.lproj** directory just contains the following two files:

〈 **InfoPlist.strings**: Its contents are shown in figure 2.



```
/* Localized versions of Info.plist keys */

CFBundleDisplayName = "Local";
```

Figure 2: The contents of the InfoPlist.strings file for the English language

CONVERT DVI TO MINIDISPLAYPORT

Gefen introduces a new solution for enabling computers with DVI connectors to utilize new Apple displays using the MiniDisplayPort connection. The converter is a low cost solution available that makes the legacy computers useful for the foreseeable future.

Gefen®
www.gefen.com
Stretch It. Switch It. Split It. Gefen's Got It.®

BUNDLED PHONE & INTERNET SERVICE

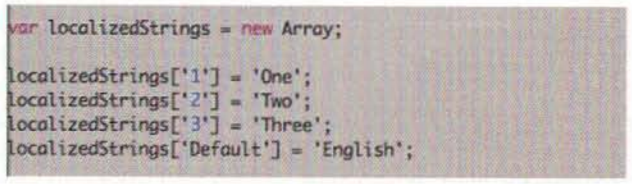
FROM $459 FLAT RATE

Dynamic Allocation T-1
Up to 16 Business Lines
Unlimited Local Service
Unlimited Site to Site Calling
2,000 Minutes of Long Distance or Toll Free

Voice Mail, Call Forwarding, 3-Way Calling, Call Hold, Pickup and Transfer, Call Waiting, Last Number Redial, DID, and DOD, Caller ID and more!

www.lowcostdialing.com
800-906-8686

⟨ **localizedStrings.js**: Its contents are shown in figure 3.



```
var localizedStrings = new Array;

localizedStrings['1'] = 'One';
localizedStrings['2'] = 'Two';
localizedStrings['3'] = 'Three';
localizedStrings['Default'] = 'English';
```

Figure 3: The contents of the localizedStrings.js file for the English language

The fr.lproj directory contents

The **fr.lproj** directory has also two files, with the same filenames as in the en.lproj directory. In this case, the strings are obviously in French :-)

The el.lproj directory contents

The el.**lproj** directory has also two files, with the same filenames as in the en.lproj directory. In this case, the strings are obviously in Greek :-)

Explaining the Technique

The key points when creating a localized Widget are to have an array that holds the localized strings as well as a JavaScript function that retrieves them.

In the Local Widget this array is called **localizedStrings** and is declared by the triplet of the **localizedStrings.js** JavaScript files (one separate file for each supported language as you already know).

Second, the JavaScript function is called **getLocalizedString** and is defined as follows:

```
function getLocalizedString(key)
{
    try
    {
        var ret = localizedStrings[key];
        if (ret == undefined)
        {
            ret = key;
        }
        return ret;
    } catch (ex) {}

    return key;
}
```

You can see by the definition of the **localizedStrings** array (see also Figure 3) that the index to it is a string (a number here but in a string manner). This has two advantages: first, it is easier to remember the index for each string, and second, if the localized string is not defined and the retrieval fails, the key string itself is returned. You are confident, then, that the JavaScript function will always return a string, no matter the

☑ Android
☑ BlackBerry
☑ Palm Pre
☑ Symbian
Sync ☑ Windows Mobile with Mac

Wirelessly

Your smartphone frees you from the land-line phone in your office. Your Bluetooth headset frees you from cabled earphones and mic. Let The Missing Sync free you from tethering your phone to your computer every time you want to sync. Sync over Wi-Fi or Bluetooth.*

Automatically

When your smartphone is near your computer, sync happens without having to think about it. Photos you snap on your phone, changes to contacts, new appointments, new music - they'll sync automatically between your phone and computer. It's just like magic!**

www.markspace.com/SyncIt

mark/space

*Wireless capabilities vary among phones, so The Missing Sync features will vary, as well.

**Proximity Sync feature not yet available for The Missing Sync for Windows Mobile.

circumstances. This is particularly important when you are designing or debugging your Widget in Safari.

You can also localize the name of your Widget–a feature that shows that Apple has covered every tiny detail when designing Dashboard. This info is located inside the **InfoPlist.strings** file that exists in each language project directory. Figure 2 presents such a file.

Figure 4 shows the Local Widget output for both the Greek and the English language. I had to change the Language Precedence List (figure 1) in order to get those two instances of the Widget. By default, I would get the English version as English is the first language in my Language Precedence List.

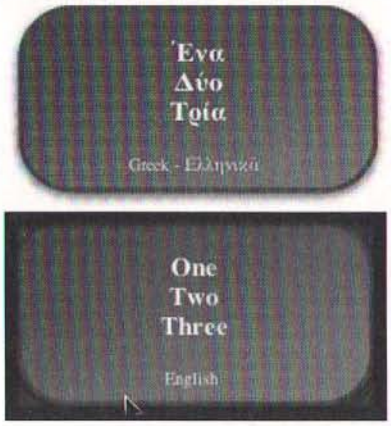


Figure 4: Two different instances (Greek and English) of the Local Widget

The full list of the Widget's files and Directories

The following is the full list (using the "*ls –aR*" UNIX command from Terminal.app) of the Widget's files and directories:

```
.:
.   Default.png  Info.plist  Local.html  el.lproj  fr.lproj
..  Icon.png     Local.css   Local.js    en.lproj
version.plist

./el.lproj:
.  ..  InfoPlist.strings  localizedStrings.js

./en.lproj:
.  ..  InfoPlist.strings  localizedStrings.js

./fr.lproj:
.  ..  InfoPlist.strings  localizedStrings.js
```

Final Advice

A timesaving technique when creating localized Widgets is to first add localization support by using Dashcode, as Dashcode makes it extremely painless to include localized strings in Dashboard Widgets.

The reason that I am telling to first start programming your Widget by adding localization in Dashcode is that you will quickly have a basic Widget skeleton to work with without being afraid that you will mess up things.

Conclusions

You should by now know how to create Widgets with localization. Localization support will add value and professionalism to your Widget.

A little exercise: you may try to add German localization to the presented Widget just to test your knowledge. It should not be too difficult!

Bibliography and References

http://www.apple.com/macosx/features/300.html - international
http://developer.apple.com/documentation/AppleApplications/Conceptual/Dashboard_ProgTopics/Articles/Localization.html

MT

About The Author

Mihalis Tsoukalos enjoys digital photography, writing articles and programming his iPhone 4 and iPad. He is the author of* Programming Dashboard Widgets, *an eBook. You can reach him at tsoukalos@sch.gr.

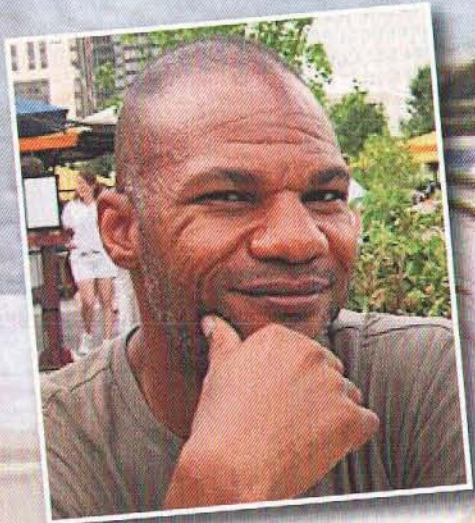
Advertiser/Product Index

Ad Index by Company for: MacTech Mag 26.08 (August/2010)

Ad Index by Product for: MacTech Mag 26.08 (August/2010)

Jayson Adams

http://www.circusponies.com



What is your company?

Circus Ponies Software, Inc.

What do you do?

I am the VP of Technology at Circus Ponies.

How long have you been doing what you do?

I have been at Circus Ponies since 2002, but I started writing apps for the NeXT platform back in 1989. I've been writing code since 1979.

What was your first computer?

It was a TRS-80, complete with Microsoft BASIC, 16k of RAM, and cassette tape storage.

Are you Mac-only, or a multi-platform person?

We're Mac and (in the future) iPad/iPhone - does that make us multi-platform? We don't support more than that.

What attracts you to working on the Mac?

The care and detail that Apple puts into making the hardware and the operating system. And the very powerful development environment that allows a few people to create applications that would take ten times that many on other platforms.

What's the coolest thing about the Mac?

I love the Dock (I run with Magnification turned on).

What is the advice you'd give to someone trying to get into this line of work today?

As far as programming goes, write, write, write. The more you write, the more familiar you'll get with the frameworks and systems, and the more you'll learn to program defensively so that you avoid your common mistakes and survive those made by others.

What's the coolest tech thing you've done using OS X?

I'd have to say a feature of our NoteBook application which is the ability to add sticky notes and flags to pages in your Notebooks. These sticky notes are rotated a few degrees from horizontal and you can edit their text at that angle (in most cases an app that displays rotated text will force you to edit the text unrotated). These sticky notes and flags can also stick out beyond the edges of your NoteBook document.

Ever?

Back in 1995 I wrote the equivalent of the Appkit in Java. At the time it looked like Java was going to be huge so I named it the Revolution Kit. I also wrote the equivalent of Interface Builder, which I named Constructor. That was the technology behind a company of mine called Netcode which we sold to Netscape in 1996, and which became known as the Internet Foundation Classes. Unfortunately it devolved after that into what's now known as the Java Foundation Classes/"Swing."

Where can we see a sample of your work?

Check out NoteBook at www.circusponies.com. NoteBook is the most popular way to get organized on the Mac. It recently won a Macworld 2010 Award from Macworld magazine in the UK.

The next way I'm going to impact the Mac universe is:

Wait 'til you see NoteBook running on the iPad.

MT

If you or someone you know belongs in the MacTech Spotlight, let us know! Send details to editorial@mactech.com

Mac shopping made easy.

Grab that to-do list, and prepare for some one-stop shopping at Smalldog.com!

Bundles simplify the buying process

Mac bundles (think Mac + RAM + AppleCare + external hard drive, etc.) not only include **everything you need,** but also **save you money.**

Visit » **Smalldog.com/specials**

Macs from under $500

We carry all **current Macs** as well as **used, refurbished and closeout models,** so there is a Mac for any budget.

Visit » **Smalldog.com/macs**

Free shipping over $200

It's true–we provide **free, same-day ground shipping** on every item over $200 every day.

Tax-free shopping

Purchases outside of Vermont are always shipped **tax-free.**

Small Dog Electronics

Always By Your Side

www.smalldog.com

800-511-MACS

Apple Specialist

Bundles

Macs

Free Shipping

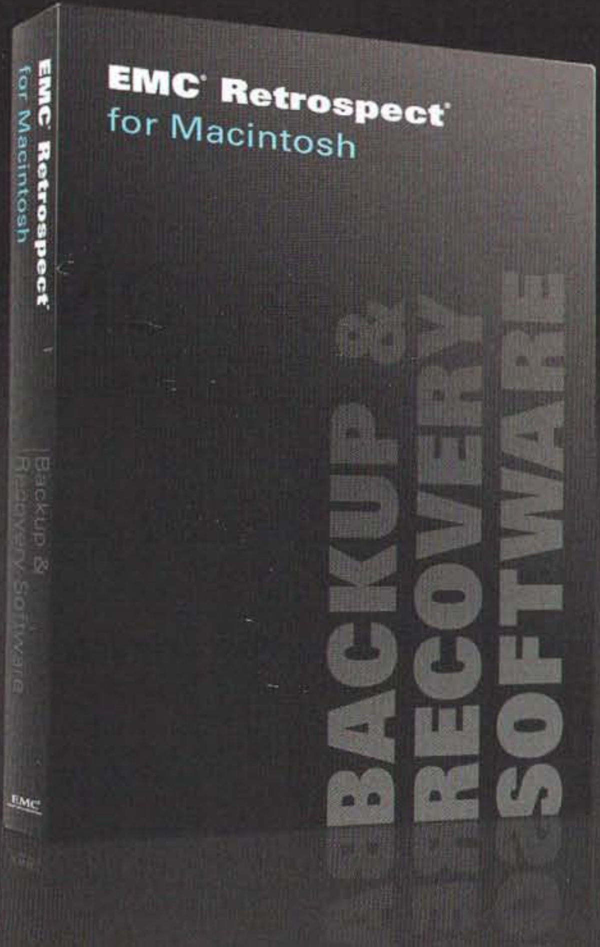
Tax-Free

Celebrating 15 Years • 3rd Largest Apple Specialist in New England • 5-Star Merchant Rating • Same-day shipping

NEW!

EMC® Retrospect® 8

backup and recovery software for small and medium businesses


EMC® Retrospect®
for Macintosh
BACKUP & RECOVERY SOFTWARE
EMC Retrospect for Macintosh
Backup & Recovery Software
EMC

The most trusted name in Mac backup

All-new EMC Retrospect 8 for Macintosh provides the reliability, ease of use, power, and flexibility you need to protect critical data on Mac and Windows PCs and servers. EMC Retrospect includes a state-of-the-art Mac user interface and enterprise-level features — including remote management of one or more backup servers, disk-to-disk-to-*anything* backups, Xsan support and custom reporting — at a fraction of the cost of other products.

Download a free 45-day trial at **www.retrospect.com/wwdc**

EMC²
where information lives®